STRATA3D: EFFICIENT AND ACCURATE AGENTIC INDOOR SCENE GENERATION WITH LAYERED FLOOR PLANS

Anonymous authors
Paper under double-blind review

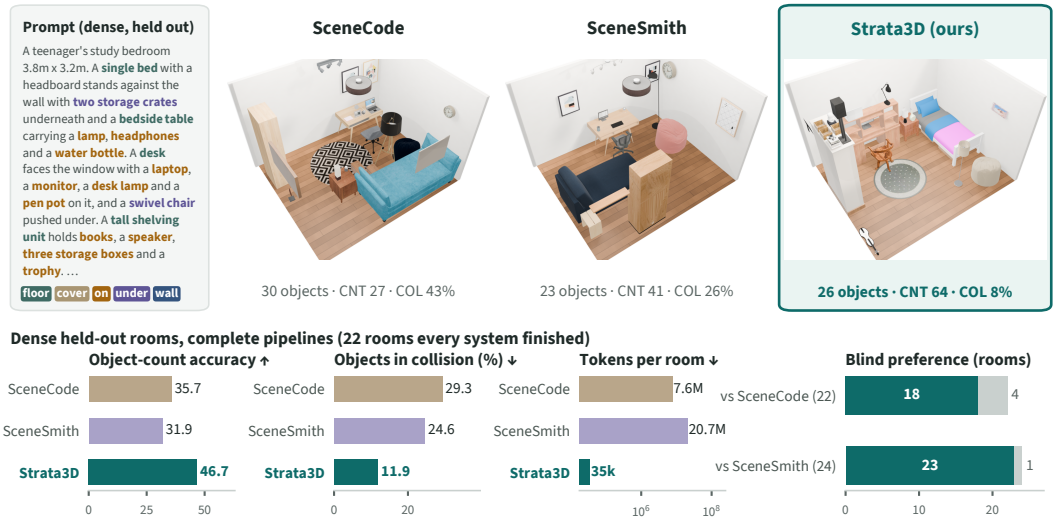

Figure 1: **Strata3D against complete agentic pipelines.** *Top:* one dense held-out prompt (object phrases coloured by the layer Strata3D's plan assigns them) given to SceneCode, SceneSmith and Strata3D; each system's own final scene, rendered identically, with that room's object count, SceneEval object-count accuracy (CNT) and collision rate (COL). *Bottom:* over the dense held-out rooms every system finished, mean object-count accuracy, share of objects in collision and median language-model tokens per room (log scale); blind pairwise preference for Strata3D over the rooms each pair was judged on.

ABSTRACT

Simulated indoor scenes are the training ground for embodied agents, and the scenes that matter most are dense: tens of objects stacked on, tucked under and hung from a few pieces of furniture. Recent agentic generators place objects directly in 3D and discover physical and functional constraints by rendering, critiquing and revising — a search that grows expensive as rooms become dense and still leaves many objects in collision. Our key insight is that many of these constraints are not three-dimensional: whether objects overlap, whether a doorway stays clear and whether every object can be reached are properties of the floor plan, *provided the plan records which objects may overlap*. We introduce Strata3D, a layout-first framework built on a *layered floor plan*, in which each object is a footprint on one of five vertical layers. The layers turn these constraints into exact, deterministic checks whose findings guide a language model in revising its plan; the validated plan is then lifted to 3D as a contract that the retrieved assets must honour. On dense prompts, against SceneCode and SceneSmith run to completion, Strata3D improves object-count accuracy by more than ten points, halves the share of colliding objects and wins 41 of 46 blind pairwise comparisons, while using two orders of magnitude fewer tokens. Ablations attribute each gain to the component that targets it, and the generated scenes can be used directly as mobile-manipulation environments.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

1 INTRODUCTION

Embodied AI has advanced by training agents in ever larger collections of simulated indoor scenes (Deitke et al., 2022; Raistrick et al., 2024; Szot et al., 2021; Li et al., 2024), and recent scene generators have adopted language-model agents that place objects directly in 3D, render, critique and revise until physical and functional checks pass (Yang et al., 2025; Ling et al., 2025; Wang et al., 2026; Pfaff et al., 2026; Xia et al., 2026). Yet this recipe hides a basic mismatch. Many of the constraints such an agent discovers by trial — whether two pieces of furniture overlap, whether a doorway stays clear, whether a person can walk to every object — are not three-dimensional; they are properties of the floor plan, decidable by a rectangle test and a flood fill before a single asset is loaded. Searching for them in 3D is expensive exactly where scenes matter most: dense rooms, where tens of objects are stacked on, tucked under and hung from a few pieces of furniture. Run to completion on such rooms, SceneCode and SceneSmith spend a median of 7.6M and 20.7M tokens per room and still leave 29.3% and 24.6% of their objects in collision. We ask whether scene generation can move this search onto the plan. Figure 1 previews the answer: deciding the planar constraints first yields rooms that follow the prompt more closely and contain fewer collisions, at a fraction of the cost.

We begin with why a plain floor plan is not enough. A flat plan is exact for most objects but wrong in precisely the configurations that dense rooms consist of: a rug under a sofa, a lamp on a desk and a chair tucked under a table all overlap in plan, and none of them collides. A checker that reads footprints at face value reports each as a collision, and a planner that obeys it moves or deletes the very objects the prompt asked for — in our ablation, only 83.9% of stacked and tucked objects stay on their host (§4.4). This motivates a representation that records how objects stack. We therefore introduce the *layered floor plan*: every object is a footprint on one of five vertical layers — on the floor, a floor covering, on a host, under a host, or on a wall — which is exactly the information needed to decide overlap on the floor plane (Figure 2).

On the layered plan, validity becomes a deterministic check. A checker tests bounds, overlap per layer, support, doorway clearance and reachability for a body of real width, and returns every violation in words, so that a language model revises its plan against facts rather than a critic's opinion. Revision loops have a characteristic failure, however: the cheapest way to silence a finding is to delete the object it names. We guard the loop against such fixes — it may not drop more than a few objects that no finding names, and it may not break the functional relations it declared — and return the best plan it has seen.

At the 3D stage, a validated plan is only useful if the scene keeps it. We therefore lift the plan under a *plan-as-contract* rule: assets are retrieved to fit each footprint and fitted to it exactly, objects are rested on their host's supporting surface, and remaining mesh contacts are repaired without leaving the planned boxes. What the checker established thus survives into the scene: before repair, 88% of the remaining mesh collisions are host–object contacts and none is a pair of floor obstacles (App. B).

Together, these components yield Strata3D, a layout-first generator that outperforms complete agentic pipelines on dense rooms. Against SceneCode and SceneSmith run to completion, it scores object-count accuracy 46.7 against 35.7 and 31.9, leaves 11.9% of objects in collision, and wins blind pairwise comparisons 18:4 against SceneCode and 23:1 against SceneSmith, at a median of 35k tokens per room.

In summary, we show that agentic scene generation can move its search from 3D to a checkable floor plan. Our contributions are:

- **Layered floor plans.** We give every footprint a vertical layer, which makes overlap, doorway clearance and body-width reachability exactly decidable on the floor plane even where objects legitimately overlap in plan.
- **Verified planning with guarded revision.** A deterministic checker states violations as facts, and a revision loop, guarded against deleting objects and breaking declared relations, uses these facts to reach a valid plan.
- **Plan-as-contract lifting.** Fit-aware retrieval, supporting-surface placement and in-box mesh repair carry the plan's guarantees into the 3D scene.
- **Controlled evaluation against complete agentic pipelines.** Under one harness and one judge, Strata3D outperforms SceneCode and SceneSmith run to completion at a fraction of their cost, and component-wise ablations of planning and lifting attribute each gain.

2 Related work

3D indoor scene generation. Early systems optimise design-guideline costs (Yu et al., 2011; Merrell et al., 2011) or sample from example arrangements (Fisher et al., 2012). Learned generators use convolutional priors (Ritchie et al., 2018), autoregressive transformers (Wang et al., 2020; Paschalidou et al., 2021) and diffusion (Tang et al., 2024; Lin & Mu, 2024), with physical guidance (Yang et al., 2024) or learned rearrangement (Wei et al., 2023), but are bounded by the categories of datasets such as 3D-FRONT (Fu et al., 2021); procedural generators (Deitke et al., 2022; Raistrick et al., 2024) trade that bound for hand-written rules. Language-model planners remove the category limit and map a prompt to a layout in a single pass: they write coordinates directly (Feng et al., 2023), state objects and relations for a search or optimiser to resolve (Yang et al., 2023; Çelen et al., 2024; Fu et al., 2023), optimise poses against vision-language scores (Sun et al., 2024), refine placements under geometric constraints (Huang et al., 2025), compose rooms from hierarchical motifs (Pun et al., 2025), or write scenes as programs (Hu et al., 2024; Aguina-Kang et al., 2024; Gumin et al., 2025a;b). These planners are largely single-pass: the language model does not inspect the finished layout and revise it, so in dense rooms, where tens of objects stack on and tuck under a few hosts, their errors stay in the scene.

Agentic 3D scene generation. The strongest recent systems close that loop with a language-model agent that places objects directly in 3D, renders the result, critiques it and revises with physical or visual feedback (Yang et al., 2025; Ling et al., 2025; Wang et al., 2026; Pfaff et al., 2026; Xia et al., 2026); follow-up work shortens their trajectories (He et al., 2026) or trains them with reachability rewards (Adhikari et al., 2026). They discover planar constraints by trial in 3D, where much of their cost goes (§4.5). Critique and external verifiers in the loop improve language-model planning (Huang et al., 2023; Kambhampati et al., 2024; Madaan et al., 2023; Shinn et al., 2023), and two recent scene systems do so: SpatialGrammar compiles bird's-eye grid placements and returns compiler errors to the agent (Tang et al., 2026), and HDSL generates and verifies subtrees of a support hierarchy (Li et al., 2026) (SAGE (Xia et al., 2026) also records a support parent per object). Strata3D keeps the agentic loop but moves it to a layered 2D plan: it verifies five vertical layers on continuous footprints, body-width reachability, tucking and declared relations before any asset is retrieved, guards the loop against deleting objects, and binds the 3D stage to the plan as a contract. Agentic systems are compared on SceneEval, which scores prompt fidelity with a vision-language judge and plausibility with geometric tests (Tam et al., 2026), or on symbolic floor-plan checks (Sengupta et al., 2026); because language-model judges are sensitive to position, verbosity and self-preference (Zheng et al., 2023; Wang et al., 2023; Panickssery et al., 2024), we report the judge-free columns separately and judge pairwise preference blind, with presentation order flipped per case.

3 Method

Our goal is to generate dense indoor scenes whose planar constraints are established on a plan rather than discovered in 3D (Figure 2). We first define the *layered floor plan*, the representation on which these constraints become decidable (§3.1). Using it, a deterministic checker states every violation as a fact (§3.2), and a guarded revision loop turns these facts into a valid plan (§3.3). Finally, the plan is lifted to 3D under a contract that the meshes must honour (§3.4).

3.1 Layered floor plans

Room and objects. Given a text prompt T, a language model returns a room Ω — a $W \times D$ rectangle with an entry segment E on its boundary, or, for a dwelling, several rectangles $\Omega_1, \ldots, \Omega_m$ in one frame joined by openings on their shared edges — and a plan $\mathcal{P} = \{b_1, \ldots, b_n\}$. Each object is

$$b_i = (R_i,\ \psi_i,\ \ell_i,\ h_i,\ t_i,\ a_i),$$

where $R_i = [x_i \pm w_i/2] \times [y_i \pm d_i/2]$ is an axis-aligned footprint in metres, ψ_i the direction the object faces (a compass heading, 0° facing $+y$), ℓ_i its layer, h_i its host, t_i a free-text description used for retrieval ("walnut dining table for six"), and a_i an optional tag naming the wall it stands against. The model emits the plan as JSON conforming to a schema that states these conventions (App. A).

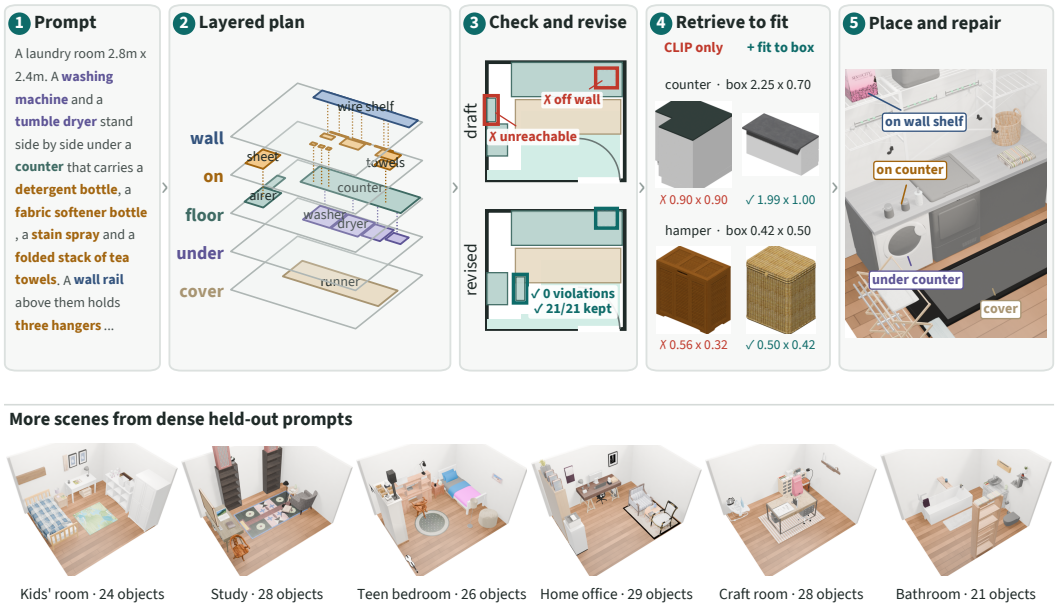

Figure 2: **Overview of Strata3D.** *Top:* one dense held-out room. (1) Prompt, coloured by layer. (2) Layered plan, exploded by layer; dotted lines join objects to hosts. (3) The checker finds two violations (red; shaded: floor reachable by a 0.55 m-wide body); revision fixes both (teal) and keeps all 21 objects. (4) Retrieval re-ranked by fit to the footprint; CLIP similarity alone picks assets of the wrong size. (5) Assets fitted to their footprints and rested on their hosts. *Bottom:* six further dense held-out rooms.

Layers. A footprint alone is not enough. It is the projection of an object onto the floor, and the projection discards exactly the information needed to decide whether two overlapping footprints collide, as the overlapping but collision-free cases of §1 show. We make the missing vertical coordinate explicit, at a coarse level. Each object takes one of five layers, which fix how it occupies the floor and where the 3D stage will put it (Table 1); on and under objects also name a host $h_i \in \mathcal{P}$. The layer is the only vertical information in the plan. It suffices for the floor-plane predicates below and leaves only vertical contact to the 3D stage: before repair, 88% of colliding mesh pairs are a host and its child (App. B).

Declared relations. With its first draft the model also declares a few functional relations $\mathcal{F}$, each $r = (\kappa, a, B)$ with $\kappa \in \{\texttt{faces}, \texttt{around}, \texttt{under}, \texttt{beside}, \texttt{near}, \texttt{flanks}\}$ ("the chairs are around the table"), each scored from the footprints as $\sigma_r \in [0, 1]$ and said to hold when $\sigma_r \geq \tau = 0.7$ (App. A). $\mathcal{F}$ is frozen after the draft, so a revision cannot satisfy geometry by abandoning an arrangement.

3.2 Plan Validation

With layers in place, validity becomes a property of the plan alone. The checker $V(\mathcal{P})$ returns a list of structured findings, each naming a predicate, the objects involved and the quantities that violate it; an empty list means the plan is accepted. The predicates are:

- **Bounds.** $R_i \subseteq \Omega$; in a dwelling, R_i lies inside the room it belongs to.
- **Overlap.** Two objects conflict if $|R_i \cap R_j| > 0.01$ m^2 and their layers put them in the same place: both floor, or both on (or both under) the same host. cover and wall objects never conflict. One exception covers tucking: a seat (e.g. chair, stool, bench) may overlap a work surface (e.g. table, desk, counter; App. A) provided the seat's centre lies outside the host's footprint — a seat centred on a table is on it, and is still reported.

Table 1: **The five layers of a plan** and what each implies for checking and for assembly.

Layer	Floor obstacle	May overlap in plan	Height in 3D
floor (bed, table)	yes	a tucked seat (see text)	on the floor
cover (rug, mat)	no, walked on	anything	on the floor
on (lamp, plate)	no	its host; not its siblings	on the host's supporting surface
under (stool, storage box)	no	its host; not its siblings	on the floor, beneath the host
wall (picture, TV)	no	anything	hung, centred at 1.55 m

- **Support.** An on or under object must lie mostly within its host: $|R_i \cap R_{h_i}| \geq \frac{1}{2}\min(|R_i|, |R_{h_i}|)$, and h_i must exist. Here overlap is the requirement, not the fault; a monitor declared on a desk but drawn beside it is reported.
- **Doorway.** No object other than a cover object intersects the door's swing, a 0.35 m-deep region inside the entry.
- **Walls and facing.** An object tagged against a wall lies within 0.15 m of it. A seat whose nearest surface (table, desk, counter) is within 0.75 m must face that surface to within 50°; the finding states the heading that would satisfy it.
- **Reachability.** Let $\rho = 0.55$ m be the body width of a person. The free space is

$$\mathcal{C} = \Omega \setminus \bigcup_{i:\ \ell_i = \text{floor}} R_i \oplus S_{\rho/2},$$

  where $S_{\rho/2}$ is a square of half-width $\rho/2$, so that $\mathcal{C}$ is the configuration space of a body of width ρ; in a dwelling, walls between rooms are solid except at their openings. We rasterise $\mathcal{C}$ at 10 cm, seed a flood fill from the cells just inside E, and call the filled set A. A floor or cover object is reachable if A has a cell within ρ of R_i; on and under objects are reached through their host, and wall objects are not tested.

The predicates are decided deterministically on axis-aligned footprints, reachability on a 10 cm grid; refining the grid to 5 and 2.5 cm changes the verdict for 1 of 689 floor objects of our final plans. None looks at a rendering, so a finding is a fact about the plan, and each carries enough to act on, e.g. "`lamp` and `vase` overlap by 0.04 m^2 on `side_table`".

3.3 Checker-guided revision

The checker says what is wrong but not how to fix it; that is left to the language model, which receives the findings in words and redraws the plan (Algorithm 1). Checker findings and unmet relations are labelled as facts to be fixed; a vision-language critic, consulted at most twice and only once the checker is silent, may add opinions on a top-down rendering, applied only where they create no finding.

Algorithm 1 Plan, check and revise. V: checker; $\mathcal{Q}$: critic; M: the language model.

$\mathcal{P}_0, \mathcal{F} \leftarrow M.\textsc{draw}(T)$; $\mathcal{P}^\star \leftarrow \mathcal{P}_0$; $K \leftarrow \emptyset$ *(keep-set)*; $c \leftarrow 0$ *(critic calls)*
for $k = 0, \ldots, 3$ **do**
  $G \leftarrow V(\mathcal{P}_k)$; $U \leftarrow \{r \in \mathcal{F} : \sigma_r(\mathcal{P}_k) < \tau\}$
  if $\mathrm{rank}(\mathcal{P}_k) > \mathrm{rank}(\mathcal{P}^\star)$ **then** $\mathcal{P}^\star \leftarrow \mathcal{P}_k$ *(rank* $= (-|G|, \sum_r \sigma_r, |\mathcal{P}|)$*)*
  if $G = \emptyset$ **then** $K \leftarrow$ names of $\mathcal{P}_k$ *(this object set provably fits)*
  $S \leftarrow \emptyset$; **if** $G = \emptyset$ **and** $c < 2$ **and** ($U \neq \emptyset$ **or** a cover object exists)
    then $S \leftarrow \mathcal{Q}(\textsc{render}(\mathcal{P}_k), T)$; $c \leftarrow c + 1$
  if $G \cup U \cup S = \emptyset$ **then return** $\mathcal{P}_k$
  $\mathcal{P}' \leftarrow M.\textsc{revise}(\mathcal{P}_k,\ \text{facts} = G \cup U,\ \text{opinions} = S,\ \text{keep} = K,\ \mathcal{F})$
  if $\mathcal{P}'$ deletes $> \max(2, \lfloor|\mathcal{P}_k|/10\rfloor)$ objects no finding names
    then ask once more; if it still does, keep $\mathcal{P}_k$
  if $\mathcal{P}'$ breaks a relation **and** $|V(\mathcal{P}')| \geq |G|$
    then ask once more; if it still does, keep $\mathcal{P}_k$
  $\mathcal{P}_{k+1} \leftarrow \mathcal{P}'$
return the better of $\mathcal{P}_4$ and $\mathcal{P}^\star$ under rank

Such a loop, however, has a characteristic failure: the cheapest way to silence a finding about an object is to delete it, and the cheapest way to satisfy a critique is to break something it did not mention. Four guards close these routes. (i) *Keep-set.* Once any round passes the checker, its object set is known to fit with clear circulation, and every later revision prompt lists exactly those objects and asks for all of them back — moved, turned or resized, but not dropped or renamed. (ii) *Deletion bound.* Enforced in code, a revision may delete objects a finding names, but deleting more than $\max(2, \lfloor n/10 \rfloor)$ others is rejected and asked again once, with the deleted names listed; a second failure keeps the previous plan. (iii) *Relation gate.* A revision that drops a declared relation below τ, or lowers its score by more than 0.25, without reducing the number of geometric findings is rejected in the same way. The loop stops when nothing is reported, or after four rounds. (iv) *Best-round return.* The loop returns the best plan it has seen under a lexicographic rank — fewest geometric findings, then highest total relation score, then most objects — so that neither a plan that deletes furniture nor a regressed last round is returned. The returned plan is not guaranteed to be valid: 96% of dense plans and all SceneEval-100 plans end with no finding, and the rest are lifted with their residual findings recorded.

3.4 From plan to scene

Retrieval. Every 3D step honours the validated plan as a contract. Each description t_i is embedded with CLIP (Radford et al., 2021) and matched against the HSSD library (Khanna et al., 2024); the 24 nearest assets are re-ranked by agreement with the planned footprint,

$$s(u) = \cos(t_i, u) - \lambda \min_{\pi} \Big(\big| \log \tfrac{w_u^\pi}{w_i} \big| + \big| \log \tfrac{d_u^\pi}{d_i} \big| \Big), \qquad \lambda = 1.2,$$

where π ranges over the asset's two horizontal alignments. For objects with a front the alignment is fixed by the planned heading rather than chosen for fit, wall objects are scored on their visible face, and candidates of implausible height for their layer are ranked last (App. A).

Assembly. The chosen asset is rotated to ψ_i first and then scaled so that its bounding box matches R_i, which makes its 3D footprint equal the validated box by construction. Heights follow the layer (Table 1). An on object rests on the host's *supporting* surface rather than its highest point: every height at which rays under the object meet the host is a candidate level, and a level is accepted only if no part of the host passes through the object, the object's centre of mass lies over the contact, and the host does not enclose it. Whether a planned under relation is possible depends on the retrieved host — a platform bed has no clearance beneath it — so each host of under objects is re-chosen down its candidate list, and if necessary re-queried as a piece "on legs", until rays cast upward through its children's footprints find room for them.

Mesh repair. A footprint does not constrain mesh geometry: a lamp's base can sink into a carved table top. After assembly every mesh pair is tested with SceneEval's contact test (Tam et al., 2026); each colliding pair moves one object (the supported child, else the wall-mounted one, else the smaller) by the smallest change that clears it without creating another: raise it (up to 6 cm), seat it on the host's top, shrink it (to no less than 0.85 of its size), or move it horizontally if the plan then yields no new finding; objects resting on it move along (App. E). The contract is thus: at assembly each footprint equals its planned box, and repair may only shrink an object inside its box or move it where the plan still validates.

4 Experiments

We first compare Strata3D with published generators on SceneEval and with complete agentic pipelines on dense rooms, where planar constraints matter most (§4.2, §4.3). Holding everything else fixed, we then remove one component at a time to attribute each gain (§4.4). Finally, we measure run-to-run variance and cost, and test the scenes as robot environments (§4.5).

4.1 Experimental setup

Benchmarks. SceneEval (Tam et al., 2026) provides 100 hand-written and 400 semi-automatic room descriptions with object, attribute and relation annotations; we report results on the 30 prompts

Table 2: **Results on SceneEval** (Tam et al., 2026), on the 30 SceneEval-100 prompts of SceneCode's evaluation; mean and 95% CI over rooms, %. †: reported by SceneCode. Right of the rule: judge-free, directly comparable (bold: best). Left: VLM-judged (GPT-4o reported, GPT-5.6 ours), directional only.

Method	CNT↑	ATR↑	OOR↑	OAR↑	SUP↑	ACC↑	NAV↑	COL↓	OOB↓	OPC↓
HSM†	53.3±9.3	58.1±14.7	22.3±12.3	50.0±10.4	70.9±7.2	85.8±4.8	99.1±1.0	18.4±6.8	7.4±3.6	10.9±8.8
LayoutVLM†	64.2±8.0	31.1±13.7	27.2±12.1	53.1±14.8	25.7±7.7	93.3±5.2	**100.0±0.0**	21.1±10.0	5.9±3.5	–
SceneSmith†	78.8±6.7	71.6±9.4	37.2±13.3	72.4±10.3	66.7±4.0	52.2±8.7	97.8±2.1	18.1±4.8	0.6±0.7	7.9±4.8
SceneCode†	79.4±6.0	74.0±11.7	32.6±12.6	61.1±12.6	44.3±12.8	74.2±9.1	**100.0±0.0**	11.3±4.3	0.4±0.9	12.4±6.7
Strata3D	80.8±8.8	78.9±9.9	64.6±13.6	72.0±13.0	91.1±5.0	82.0±6.7	**100.0±0.0**	**4.3±5.1**	**0.0±0.0**	**0.0±0.0**
same judge, the 9 rooms SceneCode finished here										
SceneCode, run here	83.3±14.2	63.3±21.7	45.7±31.1	72.2±18.2	97.2±5.4	**96.2±2.6**	98.1±2.4	8.6±9.4	**0.0±0.0**	55.6±34.4
Strata3D, same rooms	**86.1±14.7**	**78.7±13.1**	**73.2±18.7**	**80.6±24.5**	**100.0±0.0**	85.8±10.2	**100.0±0.0**	**0.0±0.0**	**0.0±0.0**	**0.0±0.0**

used by SceneCode, on SceneEval-100, and on all 500. Because these rooms are mostly sparse, we also wrote 24 *dense held-out* prompts before any system was run on them, each naming 15–30 objects, and annotated them in SceneEval-500's format (App. D).

Baselines. We run SceneCode (Wang et al., 2026) and SceneSmith (Pfaff et al., 2026) end to end (furniture, wall, ceiling and small-object stages) with Strata3D's language model (GPT-5.6) and HSSD, and score their own final scenes. We compare with SAGE (Xia et al., 2026) through 12 of its released scenes (App. I.7), and with HSM (Pun et al., 2025) and LayoutVLM (Sun et al., 2024) through SceneCode's published table.

Metrics. We use SceneEval's released code: CNT, ATR, OOR, OAR (counts, attributes, object and architecture relations), SUP, ACC, NAV, COL, OOB, OPC (support, accessibility, navigability, collisions, out-of-bounds, opening clearance). The first six use a vision-language model (VLM) judge, GPT-5.6 for every system (the original GPT-4o judge was unavailable), and are read as directional. Dense-room comparisons use the 22 rooms every system finished. Pairwise preference is judged blind (App. D). Constants are in App. A; code and scenes will be released.

4.2 Comparison on SceneEval

Table 2 compares Strata3D with published generators on SceneCode's 30 prompts. In the judge-free columns, which are directly comparable, Strata3D keeps every opening clear (OPC 0.0, against 7.9–12.4), places nothing out of bounds and leaves 4.3% of objects in collision, against 11.3–21.1%. With the judge held fixed, it also scores higher CNT than SceneCode run by us (86.1 against 83.3), though lower accessibility. The same profile holds over all 500 prompts (COL 5.8%, NAV 99.8; Table 6).

Table 3: **Comparison with complete agentic pipelines on dense held-out rooms.** SceneEval over the 22 rooms every system finished; mean and 95% CI, %; obj.: objects per room. *Baselines' furniture-stage layouts, furnished by our retrieval and exporter. Bold: best complete pipeline; OPC not comparable (baselines record no door).

Method	CNT↑	ATR↑	OOR↑	OAR↑	SUP↑	ACC↑	NAV↑	COL↓	OOB↓	OPC↓	obj.
Complete pipelines, each system's own final scenes											
SceneCode	35.7±4.7	50.9±11.4	17.9±5.2	45.0±10.2	41.5±10.3	35.8±3.9	81.7±8.0	29.3±5.0	1.6±1.5	86.4±14.7	24.7
SceneSmith	31.9±3.9	37.4±7.4	12.7±3.1	44.4±10.2	51.3±10.8	39.2±3.0	90.7±5.9	24.6±8.0	1.0±0.8	90.9±12.3	25.3
Strata3D	**46.7±7.2**	**52.3±14.9**	**31.6±8.6**	**53.1±11.1**	**75.0±7.7**	**43.4±6.5**	**96.4±3.4**	**11.9±5.3**	**0.0±0.0**	18.2±16.5	23.0
Layout-controlled: baselines' furniture-stage layouts, our retrieval and exporter*											
SceneCode, furniture stage*	24.0±4.1	42.4±11.0	5.5±2.9	51.2±10.7	90.0±5.0	80.0±7.6	87.1±7.9	19.0±8.0	0.5±1.0	63.6±20.6	6.9
SceneSmith, furniture stage*	21.5±3.4	33.5±10.4	4.9±2.6	36.1±9.5	97.4±3.2	85.6±6.1	95.5±3.4	22.2±9.7	1.3±1.8	72.7±19.0	6.5



Figure 3: **Dense held-out rooms from complete agentic pipelines and Strata3D.** Each system's own final scene, rendered with one renderer and camera (ceiling fixtures appear without a ceiling). Under each render: the number of objects placed, and that room's SceneEval object-count accuracy (CNT) and collision rate (COL).

4.3 Dense rooms against complete agentic pipelines

SceneEval metrics. Table 3 and Figure 3 compare the complete pipelines on these rooms. The baselines place a comparable number of objects to Strata3D (24.7 and 25.3 per room, against 23.0). Strata3D is better on every comparable column (all but OPC): CNT 46.7 against 35.7 and 31.9 (room-level wins:losses 15:5 and 17:3), OOR 31.6 against 17.9 and 12.7, SUP 75.0 against 41.5 and 51.3, and COL 11.9% against 29.3% and 24.6%. Deciding planar constraints first thus improves prompt fidelity and physical plausibility together.

Blind preference. In blind pairwise judging of the complete pipelines' final scenes, the automated judge prefers Strata3D 23:1 against SceneSmith (all 24 rooms; $p = 3.0 \times 10^{-6}$) and 18:4 against SceneCode (22 rooms; $p = 0.0043$; Table 8). The cross-family judge was not run on these. Against the baselines' furniture-stage layouts, both judges prefer Strata3D in every room (23:0 and 24:0; Table 8).

4.4 Ablation study

We next ask which components are responsible. SceneEval's columns are a weak probe here: removing one planning component moves them by a few points, and checking even lowers CNT and OOR, as revision trades some requested objects for validity (Table 9). Table 4 therefore measures each component by the property it targets.

Planning. *Checking and revision* make plans valid (50% → 96%) and raise the share of floor objects reachable by a person from 93.0% to 99.5%, at 3.0 rounds and about 138 s per room. *Body width* matters: checked with a point, revision leaves only 88.7% reachable, fewer than the draft. *Layers* keep the objects the prompt asked for: without them a rug under a sofa or a chair at a table is a collision, and the planner moves or deletes the stacked or tucked object — 83.9% of on and under objects stay on their host (against 94.9%), relations fall from 96.7% to 92.2%, and revision takes 4.5 rounds (re-asks included). In contrast, *relations and the critic* change none of these beyond run-to-run noise, so we attribute no gain to them. Among the revision guards (Table 10), the *deletion bound* keeps the draft's objects (in a separate revision batch, 95.2% kept without it and 98.2% with it).

Lifting. *Fitting to the plan* carries the 2D guarantees into 3D: assets at their native size overlap their boxes at IoU 78.9%, 15.9% leave the room and COL rises from 9.3% to 15.8%. *Footprint re-ranking* makes assets fit (log-size mismatch 0.29 against 1.08 with CLIP alone), though CLIP-only assets

Table 4: **Ablation study** on the dense held-out rooms; each setting removes one component and shares everything else. **(a)** Settings revise the same first draft per room; plan columns over all 24 rooms, SceneEval columns before mesh repair over the 21 every setting finished. *Valid*: no finding left; *Reach*: floor objects reachable by a 0.55 m-wide body; *Kept*/*Nested*: draft objects / draft on and under objects kept; *Rel.*: declared relations that hold. **(b)** Final plans re-lifted with one component off. *IoU*: placed footprint vs. planned box; *Size*: log-size mismatch; *Coll.*$_0$: in collision before repair; *Stable*: free objects that stay put under gravity. % except Rounds and Size. Shaded: full system; bold: best per column.

(a) Planning

Setting	Valid↑	Reach↑	Kept↑	Nested↑	Rel.↑	Rounds	CNT↑	OOR↑	COL↓
Strata3D (full planner)	96	99.5	96.2	94.9	96.7	3.0	45.4	25.9	33.1
− checker & revision (first draft)	50	93.0	**100.0**	**100.0**	97.3	0.0	48.2	**32.4**	32.9
− body width (point agent)	58	88.7	**100.0**	**100.0**	**98.3**	2.5	**48.8**	29.2	35.0
− layers (flat footprints)	**100**	**100.0**	92.1	83.9	92.2	4.5	42.9	27.5	34.8
− declared relations	**100**	**100.0**	96.8	95.8	97.3	3.0	45.6	27.0	31.8
− VLM critic	96	99.5	96.7	95.8	96.7	1.8	45.5	24.7	**31.5**

(b) Lifting

Setting	IoU↑	Size↓	Coll.$_0$ ↓	Stable↑	ATR↑	SUP↑	OOB↓	COL↓
Strata3D (full lifting)	99.6	**0.29**	31.0	63.3	62.2	77.7	**0.0**	9.3
− footprint re-ranking (CLIP only)	99.6	1.08	35.2	62.9	**66.4**	75.3	**0.0**	11.5
− fit to plan (native size)	78.9	0.30	35.1	57.3	53.9	68.5	15.9	15.8
− supporting surface (box top)	99.8	0.30	**11.2**	**64.2**	60.9	72.5	**0.0**	**6.1**
− host re-choice	99.5	**0.29**	35.0	62.7	62.1	77.7	**0.0**	10.2
− mesh repair	**100.0**	**0.29**	31.0	61.5	62.6	**82.1**	**0.0**	30.3

score higher on appearance (ATR 66.4). *Resting on the supporting surface* raises SUP from 72.5 to 77.7 at the price of more mesh contact before repair. *Mesh repair* takes COL from 30.3% to 9.3% at a small cost in support.

4.5 Robustness, cost and use as robot environments

Robustness and cost. Language-model outputs vary between runs, so we repeat the whole pipeline 4 times on the dense rooms. The runs give CNT 51.3 ± 3.3 and COL 9.7 ± 1.7 (s.d. of run means over all 24 rooms; Table 3 reports one of these runs), far smaller than the gaps to both baselines. Strata3D takes a median of 4.1 minutes and 35k tokens per dense room (tokens logged on 6 of them); the complete baselines take 7.6M and 20.7M tokens and 165 and 278 minutes for a comparable number of objects (furniture-stage costs in Figure 4).

Robot environments. A mobile Franka Panda (A* base planning, RRT-Connect (Kuffner & LaValle, 2000)) performs 114 pick-and-place tasks in 21 generated rooms (App. I.10). In rooms validated for a 0.55 m person, 35 of 49 graspable objects are within the robot's reach and 13 of the other 14 are out of reach vertically; an optional robot mode that adds the robot's reach, with an arm-reach rule, to the checker leaves 37 of 44 reachable, none vertically out of reach. A predictor reading only the plan flags 68 of the 86 failed tasks and none of the 28 successes (Figure 13).

5 Conclusion

A scene generator need not discover in 3D what can be decided on a plan. With a vertical layer on every footprint, planar constraints become exact tests, revision acts on facts, and a plan-as-contract lifting carries the result into 3D. Against complete agentic pipelines on dense rooms, Strata3D achieves higher object-count accuracy with fewer collisions, is preferred in most blind comparisons, and uses two orders of magnitude fewer tokens.

Limitations. The planar checker does not reason about height: 28% of objects resting on a host topple under gravity (App. I.8), wall-mounted hosts cannot be expressed, and most objects the robot cannot reach are too high or too low. In two rooms, revision removes many requested objects (App. A). Complete-pipeline preferences use one judge family; there is no human study.

ETHICS STATEMENT

Strata3D generates virtual indoor scenes from text for simulation and embodied-AI research. It involves no human subjects and no personal data: prompts are written descriptions of rooms, and assets come from the HSSD library under its licence, which we do not redistribute. Generated scenes could carry the biases of the language model and the asset library (for example, which furnishings count as typical for a room); we will release the prompts and plans so that such biases can be inspected.

REPRODUCIBILITY STATEMENT

The method is specified in §3 (plan, layers, checker predicates and Algorithm 1) with every constant in App. A; the evaluation protocol, judges and prompts are in App. D. We will release the code, the prompts of every set, the final plans, the scene specifications, every score file and the judged pairs with their verdicts; each number in the paper is produced by a script from these files. Language-model outputs are not deterministic; App. I.2 and §4.5 report the run-to-run spread.

USE OF LARGE LANGUAGE MODELS

Language models are part of the method and of the evaluation, as described in the paper. Separately, large language models were used for coding support and language editing. The authors take full responsibility for the content; every reported number is computed by scripts from the released results.

REFERENCES

Saugat Adhikari, Ashok Prasad Neupane, Pramish Paudel, Ajad Chhatkuli, and Danda Pani Paudel. iARCS: Iterative agentic RL for controllable 3D scene generation. *arXiv preprint arXiv:2608.06161*, 2026.

Rio Aguina-Kang, Maxim Gumin, Do Heon Han, Stewart Morris, Seung Jean Yoo, Aditya Ganeshan, R. Kenny Jones, Qiuhong Anna Wei, Kailiang Fu, and Daniel Ritchie. Open-Universe indoor scene generation using LLM program synthesis and uncurated object databases, 2024. URL https://arxiv.org/abs/2403.09675.

Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Jordi Salvador, Kiana Ehsani, Winson Han, Eric Kolve, Ali Farhadi, Aniruddha Kembhavi, and Roozbeh Mottaghi. ProcTHOR: Large-scale embodied AI using procedural generation, 2022. URL https://arxiv.org/abs/2206.06994.

Weixi Feng, Wanrong Zhu, Tsu-Jui Fu, Varun Jampani, Arjun R. Akula, Xuehai He, Sugato Basu, Xin Eric Wang, and William Yang Wang. LayoutGPT: Compositional visual planning and generation with large language models. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/3a7f9e485845dac27423375c934cb4db-Abstract-Conference.html.

Matthew Fisher, Daniel Ritchie, Manolis Savva, Thomas Funkhouser, and Pat Hanrahan. Example-based synthesis of 3D object arrangements. *ACM Transactions on Graphics*, 31(6):1–11, November 2012. ISSN 1557-7368. doi: 10.1145/2366145.2366154. URL http://dx.doi.org/10.1145/2366145.2366154.

Huan Fu, Bowen Cai, Lin Gao, Ling-Xiao Zhang, Jiaming Wang, Cao Li, Qixun Zeng, Chengyue Sun, Rongfei Jia, Binqiang Zhao, and Hao Zhang. 3D-FRONT: 3D furnished rooms with layOuts and semaNTics. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 10913–10922. IEEE, October 2021. doi: 10.1109/iccv48922.2021.01075. URL http://dx.doi.org/10.1109/iccv48922.2021.01075.

Rao Fu, Zehao Wen, Zichen Liu, and Srinath Sridhar. AnyHome: Open-vocabulary generation of structured and textured 3D homes, 2023. URL https://arxiv.org/abs/2312.06644.

Maxim Gumin, Do Heon Han, Seung Jean Yoo, Aditya Ganeshan, R. Kenny Jones, Rio Aguina-Kang, Stewart Morris, and Daniel Ritchie. Imperative vs. declarative programming paradigms for open-universe scene generation. *arXiv preprint arXiv:2504.05482*, 2025a.

Maxim Gumin, Do Heon Han, Seung Jean Yoo, Aditya Ganeshan, R. Kenny Jones, Kailiang Fu, Rio Aguina-Kang, Stewart Morris, and Daniel Ritchie. Procedural scene programs for open-universe scene generation: LLM-Free error correction via program search. *arXiv preprint arXiv:2510.16147*, 2025b.

Yun He, Kelin Yu, and Matthias Zwicker. SceneOrchestra: Efficient agentic 3D scene synthesis via full tool-call trajectory generation. *arXiv preprint arXiv:2604.19907*, 2026.

Ziniu Hu, Ahmet Iscen, Aashi Jain, Thomas Kipf, Yisong Yue, David A. Ross, Cordelia Schmid, and Alireza Fathi. SceneCraft: An LLM agent for synthesizing 3D scene as blender code, 2024. URL https://arxiv.org/abs/2403.01248.

Ian Huang, Yanan Bao, Karen Truong, Howard Zhou, Cordelia Schmid, Leonidas Guibas, and Alireza Fathi. FirePlace: Geometric refinements of LLM common sense reasoning for 3D object placement. *arXiv preprint arXiv:2503.04919*, 2025.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet, 2023. URL https://arxiv.org/abs/2310.01798.

Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas Saldyt, and Anil Murthy. LLMs can't plan, but can help planning in LLM-Modulo frameworks. 2024. doi: 10.48550/ARXIV.2402.01817. URL https://arxiv.org/abs/2402.01817.

Mukul Khanna, Yongsen Mao, Hanxiao Jiang, Sanjay Haresh, Brennan Shacklett, Dhruv Batra, Alexander Clegg, Eric Undersander, Angel X. Chang, and Manolis Savva. Habitat synthetic scenes dataset (HSSD-200): An analysis of 3D scene scale and realism tradeoffs for ObjectGoal navigation. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16384–16393. IEEE, 2024. doi: 10.1109/cvpr52733.2024.01550. URL http://dx.doi.org/10.1109/cvpr52733.2024.01550.

J.J. Kuffner and S.M. LaValle. RRT-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 2 of *ROBOT-00*, pp. 995–1001. IEEE, 2000. doi: 10.1109/robot.2000.844730. URL http://dx.doi.org/10.1109/robot.2000.844730.

Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Wensi Ai, Benjamin Martinez, Hang Yin, Michael Lingelbach, Minjune Hwang, Ayano Hiranaka, Sujay Garlanka, Arman Aydin, Sharon Lee, Jiankai Sun, Mona Anvari, Manasi Sharma, Dhruva Bansal, Samuel Hunter, Kyu-Young Kim, Alan Lou, Caleb R Matthews, Ivan Villa-Renteria, Jerry Huayang Tang, Claire Tang, Fei Xia, Yunzhu Li, Silvio Savarese, Hyowon Gweon, C. Karen Liu, Jiajun Wu, and Li Fei-Fei. BEHAVIOR-1K: A human-centered, embodied AI benchmark with 1,000 everyday activities and realistic simulation, 2024. URL https://arxiv.org/abs/2403.09227.

Letian Li, Chao Shen, Shuzhao Xie, Chenghao Gu, ZhengXiao He, Yu Meng, Xin Yang, Wenyuan Jiang, and Zhi Wang. HDSL: A hierarchical domain-specific language for structured 3D indoor scene generation and localized editing with LLM agents. *arXiv preprint arXiv:2606.09738*, 2026.

Chenguo Lin and Yadong Mu. InstructScene: Instruction-driven 3D indoor scene synthesis with semantic graph prior, 2024. URL https://arxiv.org/abs/2402.04717.

Lu Ling, Chen-Hsuan Lin, Tsung-Yi Lin, Yifan Ding, Yu Zeng, Yichen Sheng, Yunhao Ge, Ming-Yu Liu, Aniket Bera, and Zhaoshuo Li. Scenethesis: A language and vision agentic framework for 3D scene generation. *arXiv preprint arXiv:2505.02836*, 2025.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative refinement with self-feedback, 2023. URL https://arxiv.org/abs/2303.17651.

Paul Merrell, Eric Schkufza, Zeyang Li, Maneesh Agrawala, and Vladlen Koltun. Interactive furniture layout using interior design guidelines. In *ACM SIGGRAPH 2011 papers*, SIGGRAPH '11, pp. 1–10. ACM, 2011. doi: 10.1145/1964921.1964982. URL http://dx.doi.org/10.1145/1964921.1964982.

Arjun Panickssery, Samuel R. Bowman, and Shi Feng. LLM evaluators recognize and favor their own generations, 2024. URL https://arxiv.org/abs/2404.13076.

Despoina Paschalidou, Amlan Kar, Maria Shugrina, Karsten Kreis, Andreas Geiger, and Sanja Fidler. ATISS: autoregressive transformers for indoor scene synthesis. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 12013–12026, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/64986d86a17424eeac96b08a6d519059-Abstract.html.

Nicholas Pfaff, Thomas Cohn, Sergey Zakharov, Rick Cory, and Russ Tedrake. SceneSmith: Agentic generation of simulation-ready indoor scenes. *CoRR*, abs/2602.09153, 2026. URL https://doi.org/10.48550/arXiv.2602.09153.

Hou In Derek Pun, Hou In Ivan Tam, Austin T. Wang, Xiaoliang Huo, Angel X. Chang, and Manolis Savva. HSM: Hierarchical scene motifs for multi-scale indoor scene generation. *arXiv preprint arXiv:2503.16848*, 2025.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. URL https://arxiv.org/abs/2103.00020.

Alexander Raistrick, Lingjie Mei, Karhan Kayan, David Yan, Yiming Zuo, Beining Han, Hongyu Wen, Meenal Parakh, Stamatis Alexandropoulos, Lahav Lipson, Zeyu Ma, and Jia Deng. Infinigen indoors: Photorealistic indoor scenes using procedural generation. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21783–21794. IEEE, 2024. doi: 10.1109/cvpr52733.2024.02058. URL http://dx.doi.org/10.1109/cvpr52733.2024.02058.

Daniel Ritchie, Kai Wang, and Yu-an Lin. Fast and flexible indoor scene synthesis via deep convolutional generative models, 2018. URL https://arxiv.org/abs/1811.12463.

Kathakoli Sengupta, Kai Ao, and Paola Cascante-Bonilla. SceneCritic: A symbolic evaluator for 3D indoor scene synthesis. *arXiv preprint arXiv:2604.13035*, 2026.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023. URL https://arxiv.org/abs/2303.11366.

Fan-Yun Sun, Weiyu Liu, Siyi Gu, Dylan Lim, Goutam Bhat, Federico Tombari, Manling Li, Nick Haber, and Jiajun Wu. LayoutVLM: Differentiable optimization of 3D layout via vision-language models, 2024. URL https://arxiv.org/abs/2412.02193.

Andrew Szot, Alex Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Chaplot, Oleksandr Maksymets, Aaron Gokaslan, Vladimir Vondrus, Sameer Dharur, Franziska Meier, Wojciech Galuba, Angel Chang, Zsolt Kira, Vladlen Koltun, Jitendra Malik, Manolis Savva, and Dhruv Batra. Habitat 2.0: Training home assistants to rearrange their habitat, 2021. URL https://arxiv.org/abs/2106.14405.

Hou In Ivan Tam, Hou In Derek Pun, Austin T. Wang, Angel X. Chang, and Manolis Savva. SceneEval: Evaluating semantic coherence in text-conditioned 3D indoor scene synthesis. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2026.

Jiapeng Tang, Yinyu Nie, Lev Markhasin, Angela Dai, Justus Thies, and Matthias Nießner. DiffuScene: Denoising diffusion models for generative indoor scene synthesis. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pp. 20507–20518. IEEE, 2024. doi: 10.1109/CVPR52733.2024.01938. URL https://doi.org/10.1109/CVPR52733.2024.01938.

Song Tang, Kaiyong Zhao, Yuliang Li, Qingsong Yan, Penglei Sun, Junyi Zou, Qiang Wang, and Xiaowen Chu. SpatialGrammar: A domain-specific language for LLM-Based 3D indoor scene generation. *arXiv preprint arXiv:2604.27555*, 2026.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. Large language models are not fair evaluators, 2023. URL https://arxiv.org/abs/2305.17926.

Puyi Wang, Yuhao Wang, Linjie Li, Zhengyuan Yang, Kevin Qinghong Lin, Yangguang Li, and Yu Cheng. SceneCode: Executable world programs for editable indoor scenes with articulated objects. *CoRR*, abs/2605.19587, 2026. doi: 10.48550/ARXIV.2605.19587. URL https://doi.org/10.48550/arXiv.2605.19587.

Xinpeng Wang, Chandan Yeshwanth, and Matthias Nießner. SceneFormer: Indoor scene generation with transformers, 2020. URL https://arxiv.org/abs/2012.09793.

Qiuhong Anna Wei, Sijie Ding, Jeong Joon Park, Rahul Sajnani, Adrien Poulenard, Srinath Sridhar, and Leonidas Guibas. LEGO-Net: Learning regular rearrangements of objects in rooms, 2023. URL https://arxiv.org/abs/2301.09629.

Hongchi Xia, Xuan Li, Zhaoshuo Li, Qianli Ma, Jiashu Xu, Ming-Yu Liu, Yin Cui, Tsung-Yi Lin, Wei-Chiu Ma, Shenlong Wang, Shuran Song, and Fangyin Wei. SAGE: scalable agentic 3D scene generation for embodied AI. *CoRR*, abs/2602.10116, 2026. URL https://doi.org/10.48550/arXiv.2602.10116.

Yandan Yang, Baoxiong Jia, Peiyuan Zhi, and Siyuan Huang. PhyScene: Physically interactable 3D scene synthesis for embodied AI. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16262–16272. IEEE, 2024. doi: 10.1109/cvpr52733.2024.01539. URL http://dx.doi.org/10.1109/cvpr52733.2024.01539.

Yandan Yang, Baoxiong Jia, Shujie Zhang, and Siyuan Huang. SceneWeaver: All-in-one 3D scene synthesis with an extensible and self-reflective agent. *arXiv preprint arXiv:2509.20414*, 2025.

Yue Yang, Fan-Yun Sun, Luca Weihs, Eli VanderBilt, Alvaro Herrasti, Winson Han, Jiajun Wu, Nick Haber, Ranjay Krishna, Lingjie Liu, Chris Callison-Burch, Mark Yatskar, Aniruddha Kembhavi, and Christopher Clark. Holodeck: Language guided generation of 3D embodied AI environments, 2023. URL https://arxiv.org/abs/2312.09067.

Lap-Fai Yu, Sai-Kit Yeung, Chi-Keung Tang, Demetri Terzopoulos, Tony F. Chan, and Stanley J. Osher. Make it home: automatic optimization of furniture arrangement. *ACM Transactions on Graphics*, 30(4):1–12, 2011. ISSN 1557-7368. doi: 10.1145/2010324.1964981. URL http://dx.doi.org/10.1145/2010324.1964981.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-Judge with MT-Bench and chatbot arena. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/91f18a1287b398d378ef22505bf41832-Abstract-Datasets_and_Benchmarks.html.

Ata Çelen, Guo Han, Konrad Schindler, Luc Van Gool, Iro Armeni, Anton Obukhov, and Xi Wang. I-Design: Personalized LLM interior designer, 2024. URL https://arxiv.org/abs/2404.02838.

A IMPLEMENTATION DETAILS

This section lists the constants and rules behind §3. All of them are part of the code we will release; none was tuned on the evaluation prompts except where stated.

Planner. Plans are drawn and revised by GPT-5.6 through the Responses API at medium reasoning effort, with up to $24{,}000$ output tokens per call. The prompt contains the room description, the geometric rules in words (nothing may overlap or leave the room, the doorway swing stays clear, 0.55 m of walking clearance to every object) and a JSON schema that fixes the conventions: origin at the south-west corner, $+x$ east, $+y$ north, metres; footprints given by centre and extent; headings clockwise from north. The schema also defines the layers, forbids emitting windows, doors and radiators as objects, and asks that several small objects on one host be grouped by use — the plate, glass and napkin of one seat together — rather than lined up by type. A truncated or unparsable response causes the call to be retried, and rate-limited calls back off exponentially.

Relation scores. With $g(R, R')$ the edge-to-edge gap between two rectangles, $|\cdot|$ area and clip clipping to $[0, 1]$:

$$\sigma_{\texttt{under}}(a, B) = \tfrac{1}{|B|}\,\#\{b \in B : |R_a \cap R_b| > 0.15\min(|R_a|, |R_b|)\},$$
$$\sigma_{\texttt{beside}} = \sigma_{\texttt{flanks}} = \mathrm{clip}\left(1 - (\max_b g(R_a, R_b) - 0.3)/0.9\right),$$
$$\sigma_{\texttt{near}} = \mathrm{clip}\left(1 - (\min_b g(R_a, R_b) - 1.2)/2\right),$$
$$\sigma_{\texttt{around}} = \#\{\text{sides of } a \text{ occupied by } B\}/\min(4, |B|),$$

where the side an object occupies is decided by its offset from a's centre normalised by a's extents, so that a long table is judged by its shape. $\sigma_{\texttt{faces}}$ is the fraction of the shorter footprint's span that the two footprints share across the axis on which they confront. A `faces` relation declared with several targets is read as `around`: in every such declaration in our runs, the intended meaning was "these are arranged about that one". A relation holds if $\sigma_r \geq 0.7$; a revision breaks it if the score drops below 0.7 or falls by more than 0.25.

Checker. Overlaps below 0.01 m^2 are ignored. The tucking exception applies when one object's description names a seat (chair, stool, bench, ottoman) and the other's a surface (table, desk, counter, island, bar, workbench, vanity), and the seat's centre lies outside the surface's footprint. A zero-thickness door segment is given a depth of 0.35 m into the room. The facing check applies to seats whose nearest surface point lies within 0.75 m of the seat's centre and requires the heading to point within 50° of that point; its finding reports the heading that would satisfy it. In a dwelling, a 10 cm cell is floor only if it lies inside some room, and cells on a shared wall are solid unless an opening between those two rooms covers them. The flood fill is seeded from the entry segment inflated $\max(\rho, 0.3)$ m into the room; if every seed is blocked, the doorway is reported as blocked rather than every object as unreachable.

Retrieval and assembly. HSSD's CLIP index holds $14{,}099$ assets. The fit term is scored under both horizontal alignments, and the better alignment determines both the rotation and the per-axis scale; for objects with a front (seats, beds, storage, appliances, sinks, mirrors; identified by words in the name and description) the alignment implied by the planned heading is kept instead. Wall objects are scored on width and height, $|\log(w_u/w_i)| + |\log(h_u/h_i)|$. After fitting, an asset's height is scaled by the mean of its two horizontal factors, clamped to $[0.8, 1.25]$; a candidate whose resulting height exceeds 1.2 m on the on layer, 0.9 m on under or 2.6 m on any layer is ranked below every plausible candidate. Descriptions naming parts of the building (window, skylight, radiator) are skipped. For on objects the candidate support levels come from rays on a 9×9 grid over the central 88% of the object's footprint; a level is rejected if any host crossing lies strictly inside the object's vertical extent (1.2 cm tolerance), if the object's centre of mass is outside the hull of the supporting samples, or if the host covers more than half the samples from above; among the accepted levels the best-supported one is chosen, with ties within 0.10 going to the higher. If no level qualifies, the object rests on the host's top under its footprint and is marked unsupported. Wall objects are centred at 1.55 m and raised above any floor object within 0.10 m of their footprint. A host of under objects needs, under each child, a free height of 0.55 times the child's height for a seat (only the seat goes under) and the child's full height otherwise, plus 1 cm, measured by 5×5 upward rays; the first 16 ranked candidates are tried, then the queries "t on legs with open space underneath", "open frame t" and "t table".

Mesh repair. A pair collides if an FCL contact persists after the second mesh is moved 5 mm along the direction from the contact points to its centroid, as in SceneEval's COL. Raising uses 5 mm steps up to 6 cm and applies to on and wall objects; seating moves an object onto its host's highest surface if that surface is at most 1.2 m high; shrinking scales about the base centre to 0.95, 0.90 and 0.85; nudging moves a floor object away from the other in 1 cm steps up to 10 cm; a tucked object that meets its host may slide out in 5 cm steps up to 0.9 m along the four axis directions or away from the host's centre. Each candidate must clear the pair without creating a collision that the moved objects did not already have, and each horizontal move must leave the plan with no finding that it did not already have. Up to four passes are run.

Robot mode. The constants are those of the robot policy of §4.5. A surface is workable if the flood fill with $\rho = 0.74$ m (the base's diameter) reaches a cell 0.50–0.98 m from some point of the surface inset 8.5 cm from its edge. A surface at least 0.20 m wide must keep a free spot: a point 8.5 cm inside its edge and 5 cm clear of every object on it, searched on a 3 cm grid. The height prior for a description is the median height of the assets that retrieval returns for it, clamped to $[0.02, \min(3 \max(w, d), 2.4)]$ m. The grasp height of an object is its host's top plus its own height, less $\min(4.5\,\text{cm}, 0.45h)$, plus the 0.144 m flange; the arm reaches $0.89 + 0.333 + \sqrt{0.855^2 - 0.36^2} \approx 2.00$ m from the nearest stand-off at the column's highest mount. An object is flagged if its host is on a wall, if the prior makes its host 1.0 m or taller, or if its grasp height is within 0.10 m of the reach. The two thresholds (1.0 m and 0.10 m) were chosen on the 21 rooms of §4.5, where the rule flags all 14 unreachable objects and 5 of the 35 reachable ones; the plan-time experiment there therefore tests the rule as a constraint, not its generalisation.

Object retention in the main comparison. The scenes of the dense held-out rooms in Table 3 keep 541 of the 568 objects of their first drafts (95.2%): all of them in 17 of 24 rooms, one or two fewer in five, and far fewer in two (a living room, 22 → 14; a sunroom, 23 → 11).

B How exact is the plan abstraction?

The checker decides a fixed set of predicates on axis-aligned footprints; it is not a collision checker for meshes, and it decides reachability on a grid. Two audits of the final plans measure what this abstraction captures and what it leaves to the 3D stage (`pipeline/eval/abstraction_audit.py`).

Grid convergence. For every floor-standing or floor-covering object in the final plans of the dense held-out rooms and SceneEval-100 (689 objects), we decided reachability for a 0.55 m body at grid pitches of 10, 5 and 2.5 cm. The verdict changed for 1 object, which was unreached at 10 cm and reached at the finer pitches; the 10 cm grid thus erred on the conservative side.

Plan relations of mesh collisions. Before mesh repair, the final scenes of the dense held-out rooms contain 133 colliding mesh pairs in 24 rooms under SceneEval's contact test. Classified by the relation that the plan declares between the two objects, 88% are a host and the object on or under it, 2% two objects in one support stack, 2% a wall-mounted object and floor furniture, 6% a wall-mounted object and an object on a host, and 2% objects whose footprints do not overlap in the plan (a mesh extending beyond its scaled bounding box). None (0%) is a pair of floor obstacles whose footprints the checker allowed to overlap. The plan's guarantees about the floor plane therefore hold in 3D; the remaining collisions are vertical contacts that the layer abstraction deliberately permits, and these are what supporting-surface placement and mesh repair (§3.4) address.

C Lessons for evaluating scene generators

Each of the following pitfalls was observed on Strata3D; correcting each changed a reported number (Table 5).

Sampling noise sets the unit of comparison. Across identical configurations, 25.6% of 129 rooms move their furniture (median 0.51 m) while 96.9% keep an identical relation score. Counting "changed scenes" therefore measures the sampler. Comparisons need paired, semantic scores, and

Table 5: Four measurement pitfalls, each observed on Strata3D, and the effect of correcting each.

pitfall	**measurement**	**consequence**
Scenes as the unit of comparison	identical configurations, 129 rooms: 25.6% of rooms move furniture (median 0.51 m), 96.9% keep an identical relation score	"N scenes changed" measures sampling noise below 25.6%
Comparing two runs instead of two settings of one run	only a constant read after the draw changed: 0 of 40 first drafts identical	two cross-run conclusions invalidated
Judge from the planner's model family	same pairs: 7:1 same-family, 3:3 cross-family	a preference can vanish when the judge changes
Leak audit of the prompt, not the image	rendered labels `sofa` (ours) vs. `oatmeal_sofa_0` (baseline)	blinding broken although the audit passed

configurations are comparable only as settings of one run that share each room's first draft: two runs differing only in a constant read after the draw shared 0 of 40 drafts, and this confound invalidates two conclusions drawn from cross-run comparisons.

Judges and blinding. A judge from the planner's model family preferred our plans 7:1 on pairs on which a judge from another family was split 3:3; we report both families wherever both were run. A leak audit that read the prompts passed while the images carried each system's object names (`sofa` against `oatmeal_sofa_0`), and plan renders drawn without walls showed our wall-mounted objects floating. An audit must therefore cover every byte and pixel that the judge sees.

The vertical blind spot. A footprint checker is exact in the plane and silent about height. The blind spot takes two recurring forms: floor coverings that furniture stands *on* (a rug under a sofa) and objects tucked *under* another (a chair at a table). A layer-free checker counts both as collisions and inflates a baseline's violation count (App. H.1). SceneEval's accessibility metric repeats the error: because it treats every object below 2 m as a floor obstacle, it judges an object on a desk to be blocked by the desk. Metrics computed on a floor mask need each object's layer.

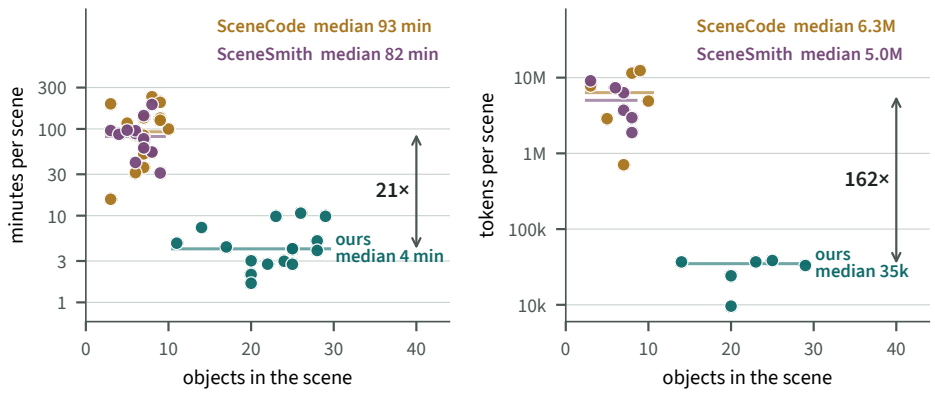

Figure 4: **Cost against scene size** on the dense held-out rooms: wall-clock minutes (left) and language-model tokens (right, rooms where every system logged tokens), log scale, one point per room and system. Baseline costs are measured up to the end of their furniture stage; the costs of the complete pipelines are given in §4.5.

Table 6: Strata3D on SceneEval-500 (500 of 500 prompts finished; IDs 0–99 form the manually written SceneEval-100, 100–499 the semi-automatic remainder), by the benchmark's difficulty label. Means over rooms; judge as in Table 2.

Prompts	n	CNT	ATR	OOR	OAR	SUP	ACC	NAV	COL	OOB	OPC
easy	150	86.1	73.9	73.6	79.8	91.1	85.1	100.0	2.2	0.0	0.7
medium	200	76.6	74.2	59.6	71.8	88.5	79.9	99.9	6.5	0.0	0.0
hard	150	63.6	57.6	44.3	57.9	82.5	68.5	99.4	8.3	0.0	3.3
all	500	75.6	68.9	58.4	69.5	87.5	78.0	99.8	5.8	0.0	1.2

D Evaluation protocol

This section describes how every result in the paper is produced: the evaluation sets and their annotations, how the baselines are run, how scenes are scored with SceneEval, how blind pairwise preference is judged, and how results are compared statistically.

D.1 Evaluation sets and annotations

We use SceneEval (Tam et al., 2026) in three subsets: the 30 SceneEval-100 prompts on which SceneCode reports results (Table 2), all of SceneEval-100, and all 500 prompts (Table 6). SceneEval scores a room against an annotation of its prompt that lists the requested objects with their counts, attributes and relations. Its first 100 annotations are written by hand and the remaining 400 are produced by a vision-language model from the text alone. The 24 dense held-out prompts were written before any system was run on them and annotated in the same way as SceneEval-500: the model is prompted with eight manual SceneEval-100 annotations, chosen by a fixed rule (the first two of each difficulty level and the two longest hard prompts), and each annotation is generated once, before any scene is scored, and never edited.

D.2 Baselines

SceneCode (Wang et al., 2026) and SceneSmith (Pfaff et al., 2026) are run from their released code through all of their stages (furniture, wall, ceiling and small objects), with the language model replaced by GPT-5.6 and the asset source set to HSSD, so that the three systems share one language model and one asset library. Each baseline is scored on its own final scene, converted to SceneEval's input format with its own meshes and transforms. A room that a system did not finish is excluded from every paired comparison involving that system. The layout-controlled comparisons of App. H instead stop each baseline after its furniture stage and furnish the resulting layout with our retrieval and exporter, so that only placement differs.

D.3 Scoring with SceneEval

We use version 1.1 of SceneEval's released code without changing any metric. Three additions connect it to the evaluated systems: an asset adapter that loads each object as the exported mesh in SceneEval's object frame, with its position and heading in the scene transform; object descriptions taken from the HSSD metadata by the rule of SceneEval's own HSSD adapter, so that no system's instance names reach the judge; and GPT-5.6 as the vision-language judge. The transform-preserving adapter matters: SceneEval reads an object's facing from its transform, and an export that bakes the geometry into room coordinates would make every object face the same direction.

Per room, CNT, ATR, OOR and OAR are the shares of the room's requirements that are satisfied; SUP is the share of objects that are supported; ACC is the mean over objects of the best functional-side score, excluding objects without a functional side; NAV is the largest connected share of free floor; COL and OOB are the shares of objects in collision and out of bounds; and OPC is the share of opening sides that are occluded. A room without requirements of a kind is omitted from that column. Tables report the mean over rooms with a 95% interval of ± 1.96 standard errors, and all rows of one comparison are paired over the rooms that every row completed. Table 6 reports Strata3D, unchanged, on all of SceneEval-500.

Table 7: Objects in mesh collision before and after the repair stage, pooled over rooms (SceneEval's test, counted by the stage itself), with the actions taken.

Variant	rooms	before	after	actions
SceneEval-100				
final: upright wall objects, steps 1–5	500	20.1%	2.9%	608
held-out 24				
registered repair (steps 1–4)	24	33.5%	9.1%	107
with slide-out (steps 1–5)	24	33.5%	5.0%	120
final: upright wall objects, steps 1–5	24	33.1%	5.0%	119
dense held-out, ablation run				
final: upright wall objects, steps 1–5	24	31.8%	6.0%	122

D.4 Blind pairwise preference

Pairwise preference compares two scenes generated from the same prompt. Both scenes are rendered by one renderer with the same camera; the presentation order is set by a deterministic per-pair flip, and the mapping from position to system is stored in a file that is read only after every verdict has been recorded. Object labels are canonicalised before rendering, since system-specific names (`sofa` against `oatmeal_sofa_0`) would reveal the system. Two judges from different model families vote independently: GPT-5.6 (the automated judge, from the planner's family) and Claude Opus 4.8 (the cross-family judge), each shown only the image pair and the request. A same-family judge can favour its own family's output (Table 5), so we report the judge family of every result and both tallies wherever both judges were run.

D.5 Statistics

Configurations of Strata3D are compared only as settings of one run that share each room's first draft, because the planner is stochastic and two separate runs share almost no drafts (App. I.2). For the same reason we compare semantic scores rather than counting how many scenes changed. Every p-value in the paper is a two-sided exact sign test over the decisive pairs; ties are reported and excluded.

E Mesh repair

Assembly fits every asset exactly to its planned box, which guarantees the plan's floor-plane properties but not that the meshes are disjoint: a lamp's base can sink into a carved table top, and a chair can meet the frame of the table it is tucked under. Mesh repair resolves such contacts after assembly. It operates on meshes, uses no language model, and never moves an object to a position the plan would reject.

Collision is decided by SceneEval's own test: two meshes collide if an FCL contact persists after the second is moved 5 mm from the contact points towards its centroid. For each colliding pair, one object moves: the child of a support relation, otherwise the wall-mounted object, otherwise the smaller one. It takes the first of the following moves that clears the pair without creating a new collision for it or for the objects resting on it, which move with it:

1. raise it in 5 mm steps up to 6 cm, if it rests on another object or on a wall;
2. seat it on its host's highest surface, if it rests on the other object of the pair;
3. shrink it about its base centre to 0.95, 0.90 or 0.85 of its size;
4. move it horizontally away from the other object in 1 cm steps up to 10 cm, if the plan then gains no finding;
5. for an object planned under its host, slide it out in 5 cm steps up to 0.9 m along the four axis directions or away from the host, re-checking the plan at each step (App. I.9).

The procedure runs for up to four passes. Table 7 reports its effect; step 5 was added after steps 1–4 and is reported separately.

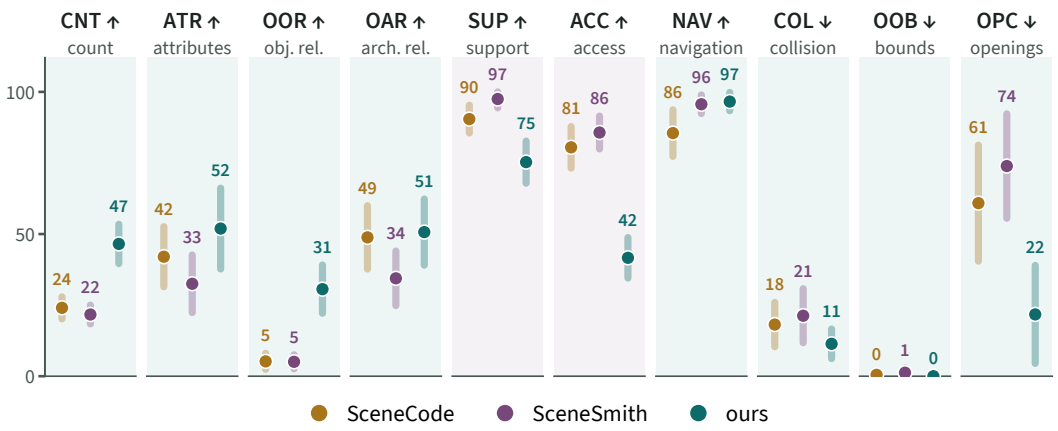

Figure 5: **SceneEval metrics on the dense held-out rooms** (mean and 95% interval), layout-controlled protocol, in which each baseline is represented by its furniture-stage layout furnished by our retrieval and exporter. OPC is not comparable, as the baselines record no door.

The exporter also stands upright any wall object whose mesh is authored lying flat (thinner than 6 cm and less than half its footprint in height), placing it against its nearest wall and facing into the room before the fit; without this rule, such a poster would be fitted as a horizontal panel (26 of 65 wall objects on SceneEval-100). The Strata3D rows of Tables 2 and 3 use this exporter and all five repair steps; the ablation settings use the earlier exporter and remain comparable with one another.

F Blind preference and layout-controlled metrics

Table 8 lists every blind pairwise comparison that we ran, and Figure 5 gives the SceneEval metrics under the layout-controlled protocol, in which the baselines' furniture-stage layouts are furnished by our retrieval and exporter so that only the layout differs.

G Component ablation

Table 9 reports the SceneEval metrics for each planning setting of Table 4(a); the rows prefixed GPT-6 repeat the settings with a second planner model, GPT-6, while all other rows and Table 4 use GPT-5.6. Table 10 ablates the guards of the revision loop (Algorithm 1) one at a time.

Table 8: Blind pairwise preference: rooms won by Strata3D, judged independently by GPT-5.6 (*auto*) and by Claude Opus 4.8 (*cross*); the mappings were opened only after both judges had voted. Each baseline is here its furniture-stage layout furnished by our retrieval, exporter and camera (the protocol marked $*$ in Table 3). *Count-matched*: the scene of Strata3D reduced to the baseline's object count by deleting objects, automated judge only. Last block: the complete pipelines' own final scenes, automated judge only (sign test $p = 0.0043$ and $p = 3.0 \times 10^{-6}$).

		full scenes		count-matched	
prompts	opponent	auto	cross	floor only	+ tucked
dense held-out (8)	SceneCode	7:0	7:0	5:2	6:1
	SceneSmith	8:0	8:0	6:2	6:2
dense held-out (16)	SceneCode	16:0	16:0	6:10	7:9
	SceneSmith	16:0	16:0	11:5	10:6
complete pipelines, each system's own final scene (§4.3)					
dense held-out (22)	SceneCode	18:4	–	–	–
dense held-out (24)	SceneSmith	23:1	–	–	–

Table 9: Component ablation. Each prompt is drawn once and every setting revises that same draw, so settings differ only in the component removed; rows are paired over the rooms every setting finished (SceneEval-100: 100, dense held-out: 21). *valid*: share of rooms whose plan passes every check of our checker (overlap, bounds, doorway, reachability for a 0.55 m-wide body, support, facing), which measures the checker against itself; $\text{NAV}_{.55}$: SceneEval's navigability with a 0.55 m agent instead of 0.2 m (dense rooms only, pre-registered as an external test of the same property); SceneEval columns in %, mean with 95% CI; obj.: objects per room.

SceneEval-100 Setting	valid↑	CNT↑	OOR↑	OAR↑	SUP↑	ACC↑	NAV↑	COL↓	$\text{NAV}_{.55}$ ↑	obj.
first draft (no checker)	66	76.4±5.1	60.5±7.7	63.5±7.5	88.8±3.0	81.1±4.4	**100.0±0.0**	12.6±4.1	–	8.0
checker + revision	100	76.6±5.0	61.1±7.7	65.3±7.5	88.9±2.8	81.1±4.3	**100.0±0.0**	12.6±4.1	–	8.0
full − VLM critic	100	77.0±5.1	60.6±7.6	65.6±7.5	88.8±3.0	81.4±4.3	**100.0±0.0**	12.6±4.1	–	8.0
full − declared relations	100	77.4±4.9	60.5±7.7	64.6±7.2	89.5±2.8	82.0±4.2	99.9±0.2	11.9±4.0	–	8.0
full − layers (flat footprints)	100	77.5±4.9	60.4±7.7	65.6±7.3	**90.5±2.5**	**82.3±4.4**	**100.0±0.0**	17.8±4.7	–	7.9
full − body width (point agent)	98	**78.0±4.9**	**62.2±7.6**	64.9±7.3	89.5±2.7	80.7±4.4	**100.0±0.1**	12.6±4.1	–	8.0
full	100	77.3±4.9	58.5±8.0	63.8±7.2	89.1±2.9	81.4±4.4	**100.0±0.0**	12.1±4.0	–	8.0
full + mesh repair (final)	100	77.9±5.0	59.5±7.7	**67.4±7.2**	89.0±2.7	80.2±4.5	**100.0±0.0**	**4.3±2.5**	–	8.0
dense held-out Setting	valid↑	CNT↑	OOR↑	OAR↑	SUP↑	ACC↑	NAV↑	COL↓	$\text{NAV}_{.55}$ ↑	obj.
first draft (no checker)	43	48.2±6.9	**32.4±7.9**	60.7±12.2	77.1±6.0	51.5±7.1	92.8±3.5	32.9±7.2	94.2	24.1
checker + revision	90	46.6±5.8	27.9±6.9	56.5±13.7	75.8±5.3	49.8±7.0	88.7±9.5	33.3±7.2	86.3	23.5
full − VLM critic	95	45.5±6.0	24.7±6.9	55.5±13.5	75.1±7.9	49.7±7.3	93.1±4.5	31.5±7.8	90.2	23.2
full − declared relations	100	45.6±6.3	27.0±5.8	**63.0±11.3**	78.9±5.6	50.1±8.6	92.0±4.9	31.8±8.2	84.3	23.2
full − layers (flat footprints)	100	42.9±6.1	27.5±6.7	53.7±11.9	**81.7±5.5**	51.9±8.0	**94.7±3.1**	34.8±8.5	91.2	22.0
full − body width (point agent)	57	**48.8±5.6**	29.2±7.4	59.2±13.7	77.7±5.6	50.0±6.5	91.4±5.5	35.0±7.6	93.3	24.1
full	95	45.4±6.3	25.9±7.5	55.7±13.1	79.8±6.1	48.8±7.6	91.8±4.3	33.1±8.3	86.4	23.1
full + mesh repair (final)	95	48.7±6.7	28.8±7.1	56.5±13.7	75.9±7.7	46.7±7.4	91.6±4.4	**10.9±5.7**	85.1	23.1
GPT-6: first draft	48	45.1±5.3	27.9±6.8	55.6±10.0	79.5±4.4	**53.0±7.1**	91.9±6.6	28.7±8.8	87.3	24.0
GPT-6: checker + revision	90	44.9±4.8	29.6±6.3	56.3±12.2	78.4±4.6	51.6±7.5	90.4±7.5	28.7±8.9	88.1	24.0
GPT-6: full	100	47.0±4.6	27.5±6.2	57.4±9.3	79.5±4.3	48.5±7.3	91.5±6.2	29.3±8.6	85.6	23.6

Table 10: **Revision-guard ablation.** The dense held-out drafts revised again in one batch with one guard of Algorithm 1 off (24 rooms); columns as in Table 4(a); bold: best in each column.

Setting	Valid↑	Reach↑	Kept↑	Nested↑	Rel.↑	Rounds
Strata3D (same batch)	**100**	**100.0**	98.2	98.1	**100.0**	1.9
− keep-set	92	99.1	99.4	**100.0**	99.2	2.0
− deletion bound	**100**	**100.0**	95.2	93.2	**100.0**	1.7
− relation gate	**100**	**100.0**	99.1	**100.0**	**100.0**	2.0
− best-round return	92	98.7	**99.6**	**100.0**	**100.0**	1.9

Table 11: Blind pairwise preference against SceneCode's furniture stage on 35 sparse rooms, automated judge (GPT-5.6), one top-down renderer for both systems. *Door drawn*: SceneCode's door is an importer's placeholder, since its exports record none; *no door*: re-judged with no door in either plan or prompt. Last row: SceneCode alone against SceneCode given our plan as its reference image, on the 12 development rooms.

	door drawn			no door		
comparison	first	second	p	first	second	p
Strata3D vs. SceneCode, by room set						
12 development rooms	12	0	0.0005	10	2	0.039
23 other rooms	16	7	0.093	16	7	0.093
pooled (35)	28	7	0.0005	26	9	0.0060
SceneCode vs. plan-informed	5	7	0.77	8	4	0.39

H Comparisons with the baselines' furniture-stage layouts

Before running the baselines to completion (§4.3), we compared Strata3D with their furniture-stage layouts under the layout-controlled protocol: each baseline's layout is furnished by our retrieval and exporter and rendered by the same renderer, so that only placement differs (App. D). This section summarises these comparisons; all preferences are blind and pairwise.

Sparse rooms. On 35 SceneEval-style rooms, twelve of which were used during development, the automated judge prefers Strata3D to SceneCode's furniture stage 26:9 ($p = 0.0060$; Table 11). The margin is significant on the twelve development rooms (10:2, $p = 0.039$) and points the same way, without significance, on the 23 other rooms (16:7, $p = 0.093$). Supplying our validated plan to SceneCode as its reference image gives no significant benefit (the plan-informed version is preferred 7:5, $p = 0.77$) and does not shorten its search (2.44 against 2.48 moves per placement). Figure 6 shows four of the development rooms.

H.1 Why the plan needs layers

A top-down projection is exact in the horizontal plane but silent about height, so two legitimate arrangements appear as overlaps: objects standing *on* a floor covering and objects tucked *under* a host (Figure 7). Applied to SceneCode's 35 layouts, a layer-free footprint test finds 67 overlapping pairs, of which at most 13 are collisions; 43 are seats tucked at a table and 11 involve a floor covering. A layer-free checker therefore misclassifies about four in five such pairs. For floor coverings the error is worse than a false positive: it forbids the footprint overlap that an `under` relation requires, and it treats rugs as obstacles when testing reachability.

H.2 Dense rooms

Dense prompts name supporting surfaces and stacked objects. We compared Strata3D with the furniture-stage layouts of SceneCode and SceneSmith on eight dense held-out prompts written before any system was run on them, and on a second set of sixteen (App. I.3). In their furniture stage, neither baseline places any object above the floor, and Strata3D places a median of 2.3–3.5 times as many objects; judging therefore uses 3D renders from one renderer and camera. Table 12 gives the results and Figure 8 shows examples.

Preference. Both judges prefer Strata3D in every room of both sets and agree on every pair. SceneCode did not finish the dense garage prompt within a six-hour limit in four attempts; we count this room as missing.

The baselines' own metrics. Under SAGE's appearance rubric, which shares the automated judge's model family, Strata3D scores higher in 7 of 7 rooms against SceneCode and 8 of 8 against SceneSmith. Strata3D also leaves a single connected floor in more rooms (7/8 and 12/16, against at most 4/7). Under the physics metrics the baselines score higher: both check physics inside their

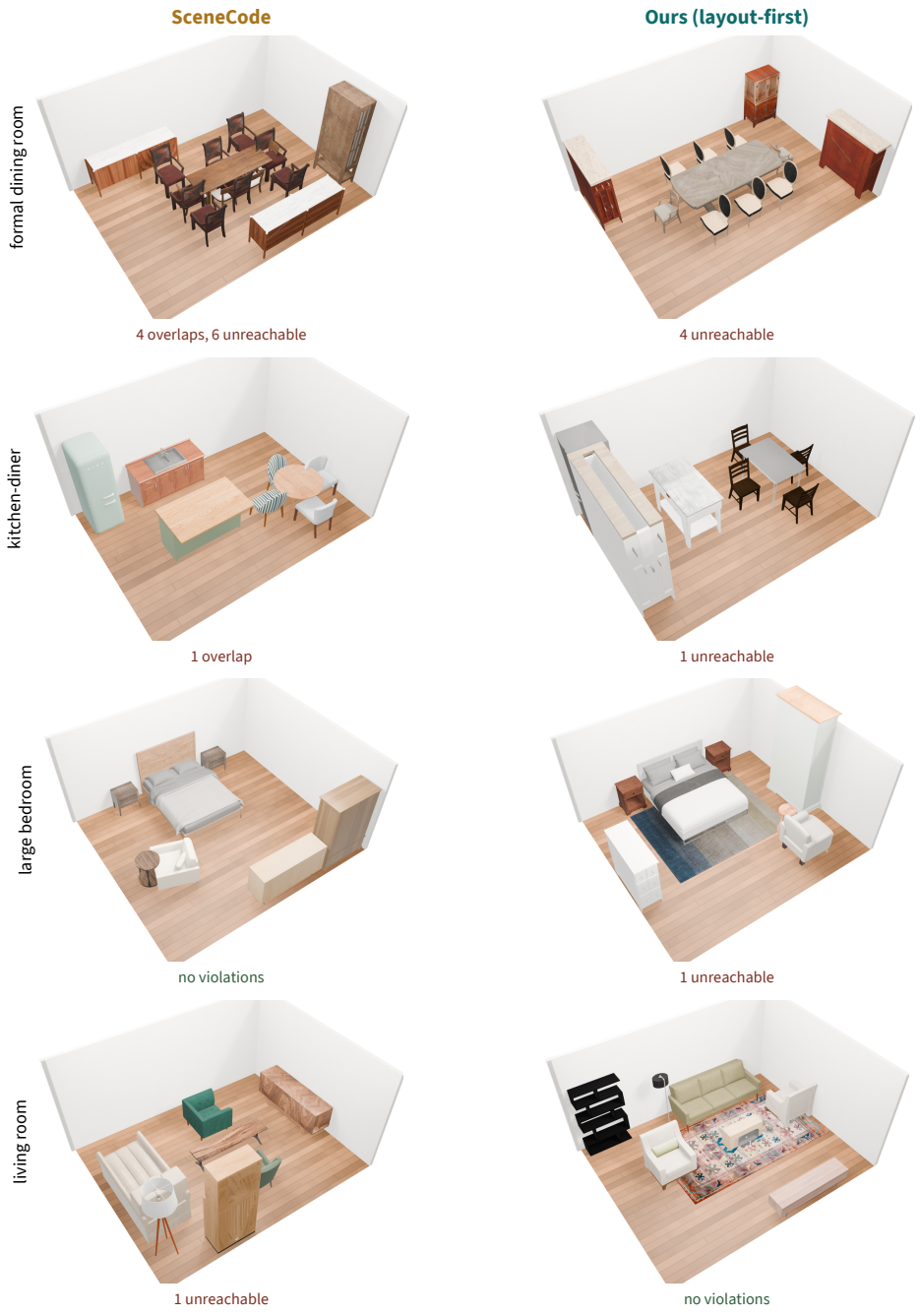

Figure 6: **Development-set rooms in 3D.** Four of the twelve development rooms, SceneCode's furniture stage (left) and Strata3D (right), both furnished by our retrieval and exporter and rendered with one camera. Counts are from the checker; in three rooms, Strata3D exhausted its revision budget before the checker reported no findings.

placement loop, whereas Strata3D fits assets into a validated plan. Every furniture–furniture collision in our scenes involves an object planned under a host whose retrieved mesh leaves no room for it, and sliding such objects clear of the host resolves most of them (App. I.9); the remaining instability lies mostly in objects stacked on others (App. I.8).

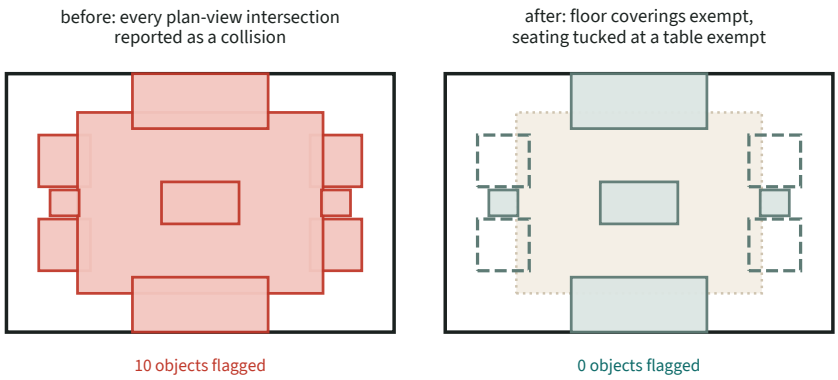

Figure 7: **Why layers are needed.** One layout checked without layers (left) and with them (right). Without layers, the rug overlaps every piece standing on it and each chair overlaps its table, so ten objects are reported as colliding; with layers, the same layout is valid.

Completeness control. Because Strata3D places more objects, a control fixed in advance keeps only our floor objects, reduced by footprint to the baseline's count. Strata3D remains preferred against both baselines on the first set (5:2 and 6:2). On the second set, the margin over SceneSmith persists (11:5) but the margin over SceneCode does not (6:10), so that margin is attributable to completeness. On requested furniture alone, SceneCode places nearly as many requested objects as Strata3D (86% against 93% on the second set).

Failure case. In one development room, a dense kitchen, the plan omits the island, its three stools and the sideboard; both judges prefer the baselines. The plan passed every check: exact validation guarantees the footprint predicates, not coverage of the request.

Table 12: Held-out results. SC: SceneCode, SS: SceneSmith; columns of Strata3D shaded. *Preference:* rooms won by Strata3D in blind pairwise judging, with every judged set re-rendered after the yaw correction (App. I.6); two judges of different families voted before the mapping was opened: *cross* (Claude Opus 4.8 with no other context, shown only the pair images and requests) and *auto* (GPT-5.6, the planner's family). The leak audit found no leak. *Metrics:* each as defined by the paper in column "def.", computed by one harness for every system (SE: SceneEval's CNT, reimplemented, App. I.5). *Cost:* per scene on one machine and one API gateway, baselines up to the end of their furniture stage; minutes over all 16 rooms, tokens (input + output; k thousand, M million) over the first 6 rooms, from each system's own usage counts.

		held-out, 8 rooms			second held-out, 16		
	def.	**ours**	**SC**	**SS**	**ours**	**SC**	**SS**
Preference: rooms won by Strata3D							
judge *cross*			7/7	8/8		16/16	16/16
judge *auto*			7/7	8/8		16/16	16/16
sign test p			0.016	0.0078		3×10^{-5}	3×10^{-5}
count-matched, *auto*, w:l (App. H.2)			5:2	6:2		6:10	11:5
Metrics as the baselines define them							
appearance rubric, 0–10 ↑	SAGE	**5.26**	2.75	2.70	**4.37**	2.94	2.53
prompt objects placed ↑	SE	**97%**	29%	20%	**91%**	28%	24%
connected share of free floor ↑	SC	**0.997**	0.952	0.888	**0.994**	0.867	0.940
rooms with one connected floor ↑	SC	**7/8**	4/7	2/8	**12/16**	8/16	7/16
objects in collision, COL ↓	SC	16.8%	**8.2%**	14.8%	10.9%	**11.7%**	16.3%
floor furniture only ↓	SC	30.2%	**9.1%**	14.8%	19.8%	**12.6%**	16.3%
static equilibrium, STB ↑	SC	31.5%	67.3%	**90.7%**	37.3%	69.4%	**80.8%**
not displaced over 0.2 m ↑	SAGE	82.1%	95.9%	**100.0%**	83.5%	98.2%	**100.0%**
Cost and scale							
minutes per scene ↓		–	–	–	**4.1**	93	82
tokens per scene ↓		–	–	–	**35 k**	6.3 M	5.0 M
objects per room		23.0	7.0	6.8	22.3	6.9	6.5



Figure 8: **Held-out rooms rendered for every system.** Three rooms from the scene files the judges saw, re-rendered with a single presentation renderer. Below each panel: object count and rubric score.



Figure 9: **Additional Strata3D scenes.** Sixteen rooms, one prompt each, generated with the final configuration and rendered with the same pipeline. All demonstration prompts are shown; none is selected on its outcome.

I Additional results

Figure 9 shows further Strata3D scenes, one per demonstration prompt.

I.1 Evaluation sets

Table 13 lists the evaluation sets used in the paper. In every pairwise comparison both sides are rendered with the same retrieval, exporter and camera. The exporter exists in two versions; the second rests objects on their host's supporting surface and bounds their height (App. I.8). All judged sets, the physics measurements and the supplementary videos use the second version.

I.2 Run-to-run variation

The planner is stochastic. Across three identical runs on 129 rooms that share one first draft per room, 25.6% of rooms differ in their furniture placement, yet the declared-relation score is identical in 125 of 129 rooms (Figure 10). Different runs thus produce different but equivalent arrangements, which is why we compare settings of one run on semantic scores (Table 5).

I.3 Second dense held-out set

Sixteen further dense prompts were written before any system was run on them; none repeats an earlier prompt. Against the furniture-stage layouts of both baselines, both judges prefer Strata3D in all sixteen rooms (16:0, $p = 3.1 \times 10^{-5}$) and agree on every pair (Table 12). Strata3D places a median of 3.4 times as many objects as SceneCode, however, and the count-matched control reverses the margin against SceneCode (6:10) while preserving it against SceneSmith (11:5). Against SceneCode on this set, the preference therefore reflects completeness rather than placement. On SAGE's appearance rubric, Strata3D scores 4.37 against 2.94 and 2.53.

I.4 Token usage

We count tokens with each system's own usage records: for Strata3D, the usage that the API returns for every planner and critic call, measured on 6 rooms of the second held-out set; for the baselines, the usage their agents log, summed over each scene (a lower bound, since calls outside an agent run are not logged). The median per scene is 35 thousand tokens for Strata3D (2–7 calls), against 6.3 million for SceneCode and 5.0 million for SceneSmith on the same rooms, measured to the end of their furniture stage (181× and 143×). Most of the baselines' usage comes from designer revision

Table 13: **Evaluation sets**: number of rooms, systems compared, what is evaluated and where it is reported.

set	rooms	compared with	what is evaluated	where
SceneEval	30, 100, 500	published generators	SceneEval metrics	Tables 2, 6
dense held-out	24	complete pipelines	SceneEval metrics, blind preference	Table 3
sparse (development and other)	12 + 23	SceneCode (furniture stage)	top-down box renders, pairwise	Table 11
dense, furniture stage	8 + 16	both baselines (furniture stage)	3D renders, pairwise; baselines' metrics	Table 12
SAGE's prompts	12	SAGE, SceneCode	3D renders, pairwise	App. I.7
four-system checker	16, 50, 38, 59	all four	our checker, each on its own prompts	Table 14
robot pick-and-place	114 tasks in 21 rooms	—	robot policy; prediction from the plan	App. I.10

rounds that re-read a context growing with the scene; Strata3D revises a JSON plan of a few thousand tokens.

I.5 Reimplemented SceneEval-style metrics

For the furniture-stage comparisons we reimplemented the SceneEval columns that can be computed on the common box format, identically for every system (the main results use SceneEval's released code, App. D.3). CNT is the share of requested objects present; OOR and OAR are the shares of stated object–object and object–room relations that hold, evaluated with fixed geometric predicates; ACC is the share of floor furniture that a 0.55 m body can reach from the door (for the baselines, which record no door, from the best door position); NAV tests whether the free floor is one connected region for a 0.3 m robot; COL is the mesh-verified collision rate and OOB the share of objects outside the room. † marks columns that our checker enforces.

set	system	CNT ↑	OOR ↑	OAR ↑	ACC ↑	NAV ↑	COL ↓	OOB ↓
held-out	Strata3D	97%	85.4	94.7	100.0†	87.5	16.8% (11.4% rep.)	0.0†
	SceneCode	29%	14.9	64.7	75.7	57.1	8.2%	1.6
	SceneSmith	20%	7.7	47.4	78.8	25.0	14.8%	1.2
second	Strata3D	91%	82.7	90.3	100.0†	75.0	10.9% (9.5% rep.)	0.0†
	SceneCode	28%	11.7	61.3	80.6	50.0	11.7%	0.0
	SceneSmith	24%	10.9	41.9	82.6	43.8	16.3%	2.6

Even from the best door position, 17–24% of the baselines' floor furniture cannot be reached. OOR favours Strata3D largely because 59% of the stated relations are "on" relations, which furniture-stage layouts cannot satisfy; excluding them, OOR is 75.6 against 43.6 and 22.2 on the held-out set and 85.4 against 35.4 and 32.9 on the second set. Figure 11 shows what each system places.

I.6 Orientation

A footprint check is invariant to a half turn, so it cannot detect an object that faces the wrong way. Three measures address orientation. First, the facing predicate of §3.2 checks that each seat near a work surface faces it. Second, the exporter converts headings from the planner's convention (clockwise from north) to the room frame of the assets; every scene and judged set in this paper is exported with this conversion, and plan-level numbers, which never read yaw, are unaffected by it. Third, objects with a front are not rotated to fit their footprint, since for a near-square seat such a rotation would turn its front away from the table. The same invariance limits the layer abstraction vertically: a footprint check says nothing about height, and a layer cannot express that an object *hangs* from its host (a tool on a pegboard), so such objects are counted separately in App. I.8.

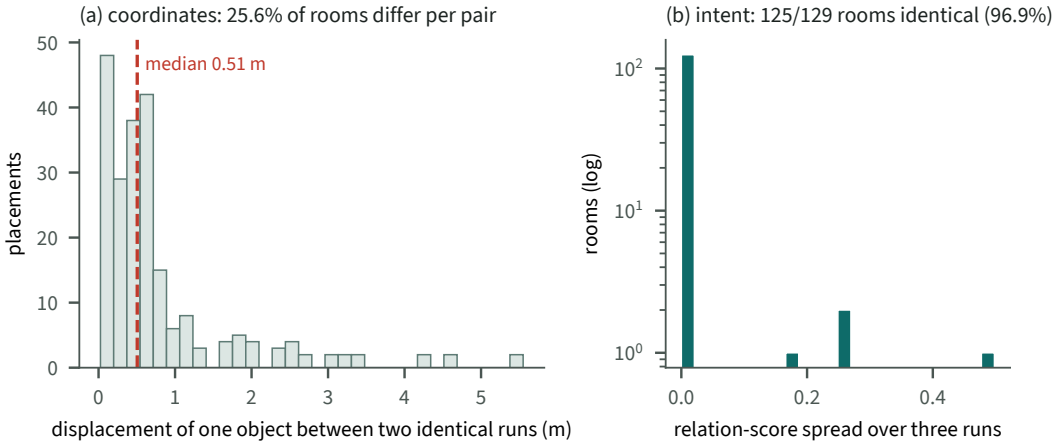

Figure 10: **Sampling noise of the planner.** Three identical configurations on 129 rooms, sharing one first draft per room. *Left:* where two identical runs disagree, furniture moves by a median of 0.51 m. *Right:* the declared-relation score is identical in 125 of 129 rooms.

I.7 SAGE

SAGE (Xia et al., 2026) generates every asset (its public release disables retrieval), whereas the other systems retrieve from one shared library; running it would conflate layout with asset generation. We therefore measure its released scenes, converted to our box format with every conversion choice favouring SAGE, and compare on its own prompts (Figure 12).

Layouts. SAGE records a support parent for every object: in 59 released scenes, 72% of 4769 objects sit above the floor. Our checker finds no overlap between floor-standing furniture in these scenes, unlike those of SceneCode and SceneSmith (Table 14). SAGE differs most in reachability: a 0.55 m body cannot reach 576 objects in 53 of 59 rooms, 205 of them furniture-sized, although SAGE lists embodied navigation as a supported use.

Same prompts. SAGE releases, for each scene, the requirement text (room size and object list) into which its planner expanded the request. Given this text, on the four densest of twelve prompts (Table 15), Strata3D places 98% of the requested objects against 48% in SAGE's released scenes, with 84% of the stated relations holding against 34%; the gap is widest for objects on surfaces, under

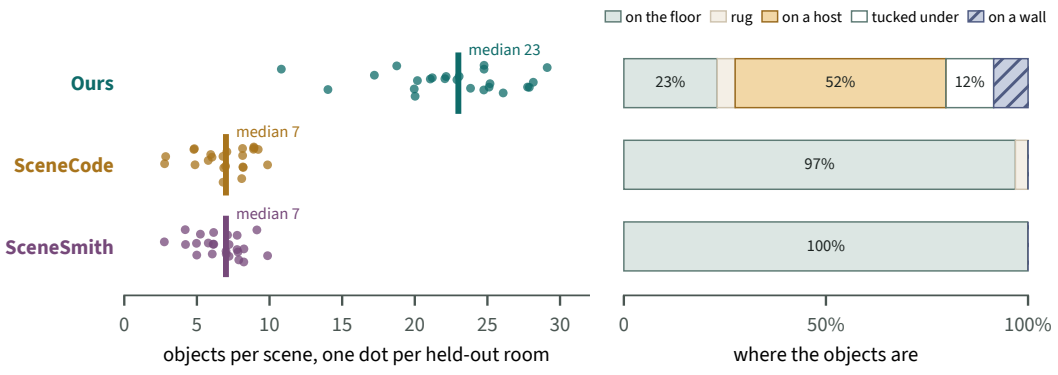

Figure 11: **What each system places** over the 24 dense held-out rooms, baselines at the end of their furniture stage. *Left:* objects per room (one dot per room, median marked). *Right:* objects by vertical layer; 72% of Strata3D's objects are on a host, under one or on a wall.

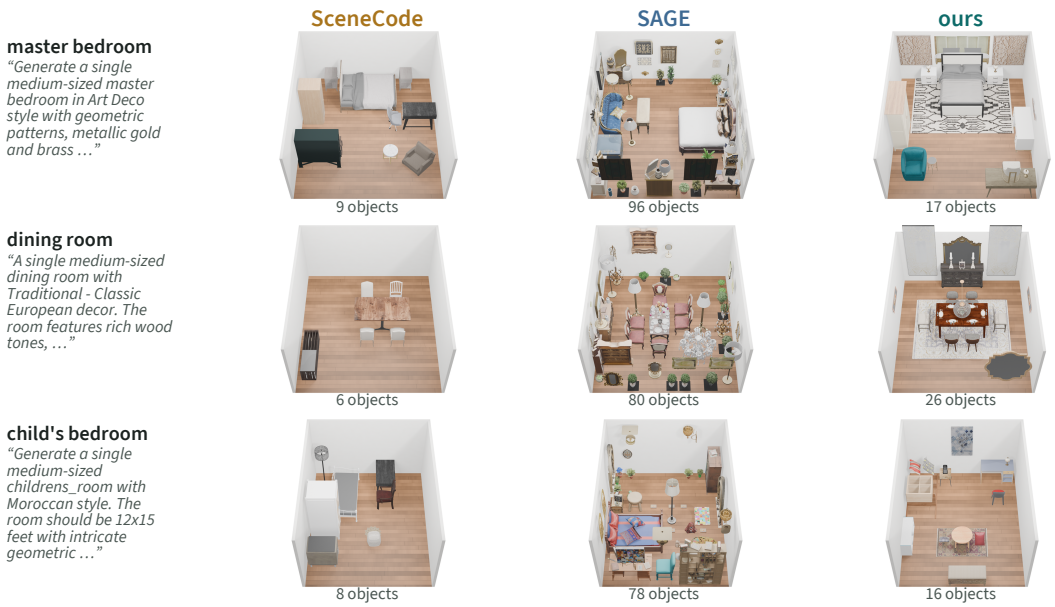

Figure 12: **Comparison on SAGE's own prompts.** The three prompts with the most objects in SAGE's released scenes: SceneCode and Strata3D given SAGE's room description, and SAGE's released scene with its own meshes, rendered with one renderer and camera. SAGE places three to six times as many objects as Strata3D, mostly small items on surfaces and walls.

furniture or on walls (98% against 44%). SAGE's generated assets are visually richer, and SceneCode has the lowest collision rate (2.9% against 10.8%).

A collapsed plan. On one dining prompt, the revision loop without the deletion bound reduced an 84-object first draft with 19 findings to 21 objects, answering findings by deleting the objects they named. With the deletion bound of §3.3, the same room keeps 84 objects and covers 72 of its 72 requested objects. This is the only such collapse among the 52 runs that record per-round object counts. The main comparison was produced without this guard; its effect is measured in Table 10.

Blind preference on SAGE's prompts. On twelve of SAGE's rooms, Strata3D was run on SAGE's short style text (setting A) and on SAGE's object list (setting B). The automated judge prefers Strata3D 10:2 in both settings ($p = 0.04$), and the cross-family judge 10:2 and 9:3. The comparison passes SAGE's layouts through our conversion, and SAGE's scenes contain more objects. Against SceneCode on the same prompts, the automated judge gives 9:3 ($p = 0.15$) and the cross-family judge 10:2 ($p = 0.04$).

I.8 Physical stability

Final system. We settle each scene under gravity in Drake and call an object stable if it moves less than 5 cm and turns less than 0.3 rad. On the final exports of the 24 dense held-out rooms, 87% of 134 floor objects are stable and 2% topple, and 79% of tucked objects are stable. Objects resting on a host are the weak layer: 51% of 251 are stable and 28% topple. Objects hung from a wall-mounted host, which the layers cannot express, mostly fall (62%).

Harness. Each asset is decomposed into convex parts (CoACD) that are placed by an exact affine fit to the exported mesh; mass and inertia are computed from the parts. Floor furniture and on and under objects are free bodies, and wall and cover objects are fixed. Because convex parts bulge beyond the mesh, each body may rise by its own bulge plus 5 mm before settling, and pairs that still overlap are not allowed to collide with each other, so that an initial overlap does not knock over neighbouring objects. The same harness revealed defects of the lifting step that the final system fixes,

Table 14: One checker, four systems, each on its own layouts and prompts (not a head-to-head comparison: SAGE's rooms hold a median of 77 objects, SceneCode's and SceneSmith's about 8). Overlaps are net of the tuck-under and floor-covering exemptions. The baselines record no door, so their reachability is scored from the best door position.

	rooms	objects	above floor	clean	overlaps	*of those, floor*	unreachable
Strata3D	16	395	269 (68%)	16/16	0	0	0
SceneCode	50	379	0 (0%)	10/50	38	38	69
SceneSmith	38	297	0 (0%)	8/38	34	34	56
SAGE	59	4769	3450 (72%)	2/59	478	0	576

among them objects resting on a host's bounding box rather than its surface, unbounded heights, and under objects were placed inside hosts with no room beneath them (§3.4).

Comparison with the furniture-stage baselines. Under SceneCode's definition (stable if moving less than 1 cm and turning less than 0.1 rad), on the first held-out set:

	free	floor stable	floor toppled	stacked	stacked toppled	started inside
Strata3D	163	44.0%	12.0%	71	26.8%	55 (31)
SceneCode	49	70.0%	3.3%	–	–	19 (0)
SceneSmith	54	97.2%	0.0%	–	–	18 (0)

Floor percentages are over objects that start without contact; *started inside* counts objects that begin overlapping another (in parentheses, overlapping only in the convex proxy). The floor columns are comparable; the stacked columns exist only for Strata3D, since the furniture-stage baselines place nothing above the floor. Roughly a quarter of our stacked objects that start clean still topple, and we have not identified the cause.

I.9 Repairing under-host collisions

Every furniture collision in our furniture-stage comparison involves an object planned under a host whose retrieved mesh leaves no room for it. Sliding each such object out in 5 cm steps (up to 0.9 m), keeping a step only if the plan remains valid and the meshes are clear, repairs 7 of 8 such objects in the first held-out set and 9 of 13 in the second, and reduces COL from 16.8% to 11.4% and from 10.9% to 9.5%. Stability barely changes (STB 29.3% and 37.5%), since it is limited by stacked objects, which the repair does not move. This step is part of the mesh repair of App. E.

I.10 Robot environments

Setup. A Franka Panda on a holonomic base with a lifting column performs pick-and-place tasks in 21 generated rooms (A* base planning, inverse kinematics for top-down grasps, RRT-Connect for arm motions, collision checking in Drake). The 114 tasks pair every graspable object with every other occupied surface.

Predicting failures from the plan. A predictor that reads only the lifted plan (boxes, heights, layers, hosts) and applies the robot's geometric limits (base clearance, stand-off distance, arm reach, free placement spot) was fixed before any task ran. The robot succeeds in 28 of 114 tasks; the predictor flags 68 of the 86 failures and none of the successes (precision 1.00, recall 0.79), meeting the criterion set in advance. The missed failures are arm-path failures after a grasp, objects that slide after release, and placement spots the policy failed to find. The predictor encodes this policy's rules, so the result shows that its constraints are decidable from the plan, not that another robot would fail in the same places (Figure 13).

Robot mode. The same rules, without heights, can run inside the checker. With the robot's base width and stand-off, robot mode clears every such finding at 21.7 objects per room (against 22.0) and raises task success from 16 of 116 to 24 of 113 (13.8% to 21.2%), reducing failures at the place

Table 15: **Four systems on SAGE's own prompts.** SAGE was not run (App. I.7); its entry is its released scene, with its own generated meshes; the other three systems were given SAGE's requirement text verbatim (room size and object list) and reach the renderer through one retrieval, exporter and camera. The four rooms are the densest of the twelve four-way prompts by requested-object count (65–77 objects), fixed before any run. Prompt recall is the share of requested objects placed and relations the share of stated relations that hold; *on* counts direct support only. *Ours (first draft)* is the run as registered; *Ours* is the system after the four changes that this run motivated (App. I.7): the revision guard, place settings, wall height, and fronts kept when fitting an asset. SAGE was not run, so it has no wall-clock or token cost.

	objects	recall	furniture	on, under, on walls	relations	COL	minutes
Ours	318	98%	100%	98%	84%	10.8%	4.9
Ours (first draft)	257	81%	99%	75%	64%	6.2%	3.5
SAGE (released)	258	48%	81%	44%	34%	13.0%	—
SceneCode	40	14%	100%	0%	1%	2.9%	34
SceneSmith	40	12%	84%	0%	0%	11.8%	68

from 23 to 13. Failures at the pick barely change (60 to 56) because they are mostly vertical: of 49 graspable objects in rooms validated for a 0.55 m person, 35 are within the robot's reach, and 13 of the other 14 are out of reach vertically. Adding a plan-time arm-reach rule leaves 37 of 44 objects reachable, none of them out of reach vertically. A post-lift height check, by contrast, has no effect on task success (23 of 113 against 24 of 113), because the remaining pick failures stem from the policy's own base-pose search.

I.11 Editing a validated plan

Our plan can be edited through operations (add, move, resize or remove a box) applied by code, so an object that a request does not name moves only if the revision loop must move it. We compared this with rewriting the whole plan, as two settings of one run, on the 16 second-set plans with one add, one move and one remove request per room. An unnamed object was moved in 3 of 48 operation edits and in 11 of 48 rewrites (paired 8:0, $p = 0.0078$); the result replicates on a second request set (12:1, $p = 0.0034$). Operations complete fewer moves (40 of 48 requests against 44), partly because

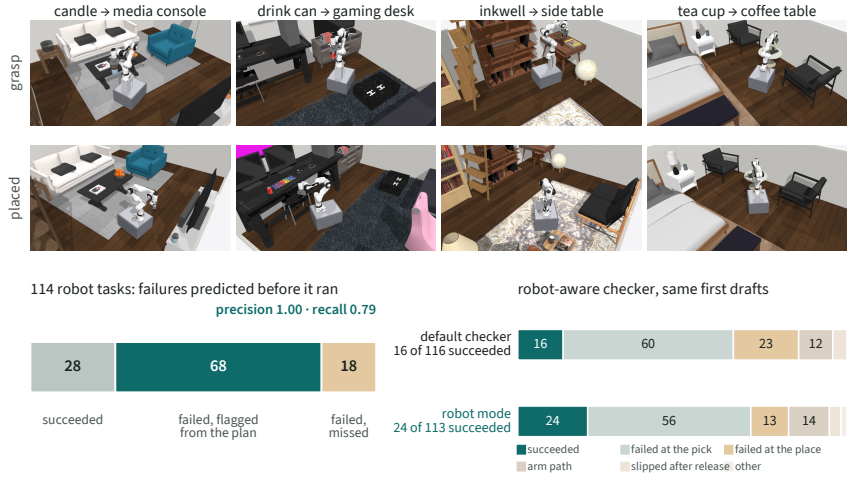

Figure 13: **Strata3D rooms as robot environments.** *Top:* four pick-and-place tasks executed by a Franka Panda on a holonomic base, shown at the grasp and after the place. *Bottom left:* a predictor that reads only the plan flags 68 of the 86 failed tasks and none of the 28 successes. *Bottom right:* task outcomes with and without robot mode.

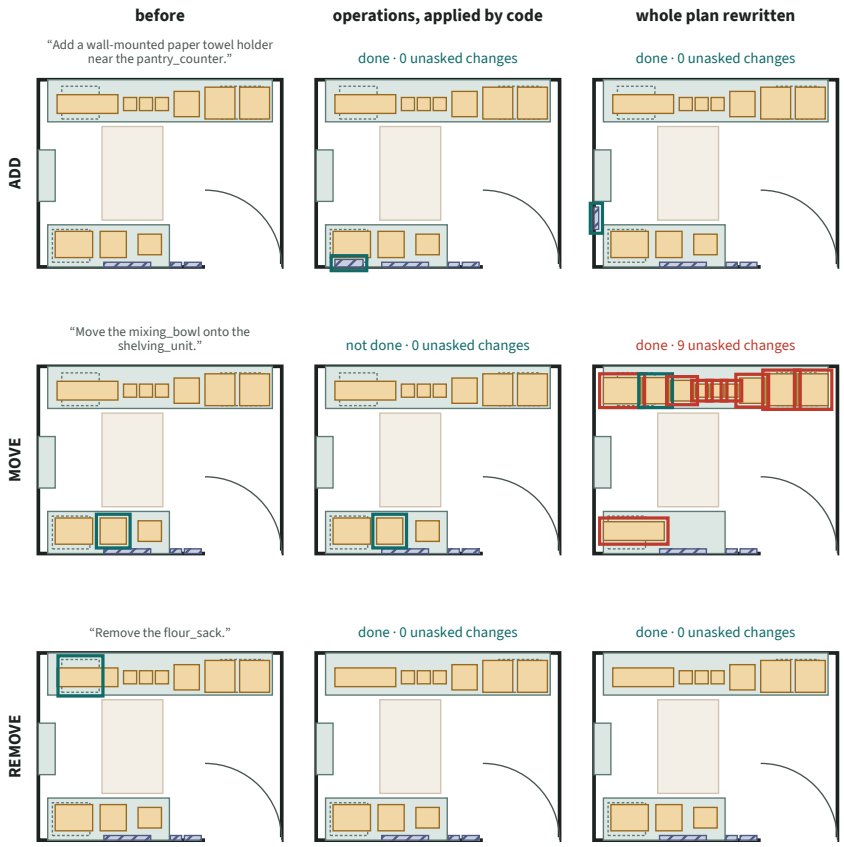

Figure 14: **Editing a validated plan.** Three requests on one pantry plan (rows: add, move, remove). Columns: the plan before the edit, the edit applied as operations by code, and the whole plan rewritten by the model. Teal: the object the request names or adds; red: other objects the edit moved.

a move cannot change an object's host; allowing a move to set the host completes two more of them without additional unrequested changes. Figure 14 shows an example.

I.12 Supplementary material

The supplementary material contains, for each of the eight held-out rooms, a turntable of the Strata3D scene and a synchronised three-way clip of Strata3D, SceneCode and SceneSmith rendered with one camera path; physics replays for three rooms; six multi-room tours; and a web page that loads a room from any of the three systems with a gravity switch.